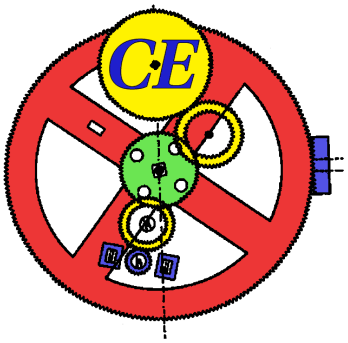


Computer Engineering
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ce.et.tudelft.nl/

2009

MSc THESIS

On-line Testing of Routers in Networks-on-Chips

Ghazaleh Nazarian

CE-MS-2009-18

Abstract

Due to recent progress in semiconductor technology, communication is becoming the major source of expense in today's chip design. Network-on-Chip (NoC) is a new paradigm for solving the problem of complex communication on the chips. However, in order to NoC to be efficient in providing complex on-chip communication, the designers should be assured about its correct functionality. For this reason, an efficient test method should be developed for testing the NoC. In order to test the NoC in an efficient way, the test method should be in a way that does not degrade the performance of the NoC (with online test). Moreover it should test the NoC with a high fault coverage (structural test). In this thesis, a new platform is proposed to do online-structural test on NoC. Although NoC's elements has been tested after manufacturing and before being used in a SoC, but when NoC is being used, after some time there is possibility that some errors occur in the elements (e.g. router) of NoC and ruin their functionality. The proposed platform, helps the system developers with online detection and localization of the errors that may occur in routers. Therefore, it increases NoC's reliability. In this platform, the idle routers of NoC are being tested while the rest of the routers are providing on-chip communication. For testing the router the standard test architecture for testing embedded cores (Test-Access-Mechanism (TAM), wrapper, source and sink)is used. In this architecture it is assumed that *source* and *sink* are given. The main contribution of this project is to design the wrapper element in such a way that it can be used in the proposed online-structural test platform and reuse NoC as TAM. The limitation of the proposed method is that the number of routers that can be tested at the same time without interrupting NoC's normal functionality depends on topology and size of the NoC. The major cost of this test methods, is related to the wrapper architecture.

TU Delft

Faculty of Electrical Engineering, Mathematics and Computer Science

On-line Testing of Routers in Networks-on-Chips

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Ghazaleh Nazarian
born in Tehran, Iran

Computer Engineering
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

On-line Testing of Routers in Networks-on-Chips

by Ghazaleh Nazarian

Abstract

Due to recent progress in semiconductor technology, communication is becoming the major source of expense in today's chip design. Network-on-Chip (NoC) is a new paradigm for solving the problem of complex communication on the chips. However, in order to NoC to be efficient in providing complex on-chip communication, the designers should be assured about its correct functionality. For this reason, an efficient test method should be developed for testing the NoC. In order to test the NoC in an efficient way, the test method should be in a way that does not degrade the performance of the NoC (with online test). Moreover it should test the NoC with a high fault coverage (structural test). In this thesis, a new platform is proposed to do online-structural test on NoC. Although NoC's elements has been tested after manufacturing and before being used in a SoC, but when NoC is being used, after some time there is possibility that some errors occur in the elements (e.g. router) of NoC and ruin their functionality. The proposed platform, helps the system developers with online detection and localization of the errors that may occur in routers. Therefore, it increases NoC's reliability. In this platform, the idle routers of NoC are being tested while the rest of the routers are providing on-chip communication. For testing the router the standard test architecture for testing embedded cores (Test-Access-Mechanism (TAM), wrapper, source and sink)is used. In this architecture it is assumed that *source* and *sink* are given. The main contribution of this project is to design the wrapper element in such a way that it can be used in the proposed online-structural test platform and reuse NoC as TAM. The limitation of the proposed method is that the number of routers that can be tested at the same time without interrupting NoC's normal functionality depends on topology and size of the NoC. The major cost of this test methods, is related to the wrapper architecture.

Laboratory : Computer Engineering
Codenumber : CE-MS-2009-18

Committee Members :

Advisor:	Said Hamdioui
Member:	Said Hamdioui
Member:	Kees Goossens
Member:	Nick van der Meijs

In the memory of my mother (Fami Navabi) and
To my dear father (Iradj Nazarian).

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my Adviser Said Hamdioui for the patience and encouragement in the process of developing this project.

I also want to thank Professor Kees Goossens who has given me advice and technical support on the AEthereal NoC design tools.

Finally, I would like to acknowledge the support of all CE members, my fellow students and my family.

Ghazaleh Nazarian
Delft, The Netherlands
February 3, 2009

1 Introduction

Today's chip design industry witnesses some new challenges that in earlier times, it had not to be worried about. These new challenges are basically due to technology down-scaling. *Transistors*, the basic composing elements of chips, are decreasing in size and increasing in number. This is demonstrated by the *Moore's law* in 1965. The Moore's law states that every 18 month the number of transistors and thus hardware blocks doubles on the chip. Due to this incident the density of transistors and hardware blocks are increasing on the chip exponentially. As a consequence , in today's chip design, communication between on-chip cores is becoming more expensive than on-chip computation [5]. In other words, providing an interconnect system for the chip is becoming more costly than on-chip blocks. Therefore, the designers has proposed to exchange the conventional on-chip interconnect system (buses) with a new on-chip interconnect, named as *Network on Chips (NoC)*.

Special characteristics of NoC (such as scalability, being modular, flexibility, etc) makes it a perfect choice for today's on-chip communication. But, because of critical responsibility of NoC in today's chips, its testing becomes a crucial task.

1.1 Current router test methods and their shortcomings

In order to test NoC, its three main components (routers, network interfaces and links) have to be tested. Since routers have a critical responsibility in NoC's functionality, in this thesis the focus is on router testing.

Routers of an NoC can be seen as on-chip cores. Therefore, the same style as that for embedded core testing can be reused to test NoC routers. In order to test an embedded core on the chip, a special test architecture should be used. This test architecture is composed of test generator, test analyzer, a mechanism to transport test data to/from the embedded core and a wrapper around the core-under-test. State of the art methods for testing the routers of NoC use this test architecture for testing routers.

From different aspects of testing, testing NoC can be done either *functional* versus *structural* or *online* versus *offline*.

Online testing a device makes it possible to test the device without interrupting its normal functionality. Structural testing, gives us the opportunity to locate the exact place of defect.

State of the art methods of NoC testing are using offline-structural or online-functional testing. However, due to the critical responsibility of NoC and its special characteristics, online-structural testing is a promising test method in NoC. Because, in addition to being tested online and without interruption of on-chip connection, it is possible to localize the faults for diagnosis.

1.2 Contribution of this project

In this project, it is determined why the online-structural test can be beneficial. A new platform for online-structural test of routers in NoC is proposed. This allows testing of the routers, while the NoC is performing its normal functionality. The normal functionality of the NoC is to transmit the data packets between the on-chip IP cores. The purpose is to test the routers with the least impact on the normal functionality of the NoC. In the final platform, concurrently when an application is running on the SoC and NoC is providing connections between the on-chip cores, the routers of the NoC can also be tested. It is assumed that network interfaces and links of the NoC has been tested before and are fault-free.

For testing the routers in the proposed platform, they are considered as on-chip cores. It consists of four basic elements namely test pattern generator, test response analyzer, test access mechanism and a wrapper. The test generator (source) and test analyzer (sink) are assumed to be available. In order to transport test data to/from routers-under-test, the NoC itself is being reused. The architecture that should wrap around the router (wrapper), is the part that should be designed. The main contribution of this work is to design a wrapper with specific characteristics that can be used in the proposed platform for online-structural NoC test.

The target NoC in this project that the wrappers are designed for its routers, is AEthereal NoC.

1.3 Outline of the report

The rest of the report is organized as follow. In chapter 2, the background information on NoC design and testing methods is given. It is discussed why NoC is needed and why it should be tested. Moreover the characteristics and architecture of NoC and different methods of testing are given.

In chapter 3, the previous researches about the NoC testing are introduced. It discusses about researches on testing different parts of NoC (links, routers, network interfaces). Moreover methods used in NoC for making increasing its reliability are discussed.

Chapter 4 introduces the proposed method for online-structural testing of NoC. The reason for online-structural test in NoC is explained. The requirements for realizing this method of test is defined and finally the required works is formulated. The two major requirements for this method are designing a wrapper for routers and a high-level scheduler. In this project, the first one (designing the wrapper) is addressed.

In chapter 5, the implementation of the wrapper is explained and its simulation results are discussed. Finally, last chapter gives the conclusions and state the related subjects for future research.

2 NoC design and test

In today's new technology, size of transistors is getting smaller in a way that make it possible to integrate millions of transistors on a single chip. *Integrated Circuits (IC)* are composed of a layer especially for placing transistors, and a number of layers that wires are placed on them (known as metal layers). Wires connecting different layers together are known as via. Local wires on the chip are thin and short; however we need to use fat long wires for global communications.

In 1965, Gordon E. Moore observed that the number of transistors, which is efficiently exploited in a single chip, is increasing exponentially (doubles almost per two years). The effect of Moore's law on chips is that from one process generation to the next, the number of transistors increases and they get smaller and faster. In order to not degrade the chip speed due to inter transistor connections, we should use the same number and size for global wires as it was used in previous process generations. The problem with using long wires is that they require buffers (inverters) for amplifying attenuated signal due to physical effects, like resistance of wires. Thus, in addition to large number of wires that should be used for connecting transistors in metal layers, buffers that are used for amplifying waste an extra space also in transistor layer of IC. Therefore in today's chip design as computation is becoming cheaper, communication is getting more expensive than computation. This issue necessitates communication-centric designs and focusing on using a proper scheme for the on-chip interconnects in order to reduce the cost of designs.

If we look at the problem of on-chip interconnection from a higher level point of view, in a *System on Chip (SoC)* there are several *Intellectual Property (IP)* blocks connected to each other. Different solutions of on-chip interconnections are: shared buses and *Network-on-Chip (NoC)*. Two IP cores (master and slave) can connect to each other either by message passing (streaming data) or by reading and writing from a shared memory. Connection between them is realized by request and response transactions which takes place on IP ports and is transferred through the interconnect. Latency of transactions depends on the level of concurrency that the interconnect can provide. Level of concurrency is defined by whether transactions are splitting (request and response can be split) or not, whether they are pipelined or not and whether connections are multi layered (using threads or virtual channels).

Buses are old conventional methods with simple structures. They have the lowest degree of concurrency. The connection between IP cores that are connected by bus by transmitting transactions as message. If there are multiple transactions aiming to access the bus, there must be an arbitration mechanism to choose between them. Therefore by increasing number of IP cores on chip, the time for accessing the bus for each IP core will increase [1].

Performance of on-chip interconnect is defined by minimum latency and maximum

bandwidth (throughput) that interconnect can provide for connections between master and slave. But, as explained buses do not have adequate performance in today's complex designs. NoC has the highest level of parallelism leading to smaller latency when number of connections increase [18]. Therefore NoC solve problems of performance loss in buses which is mainly due to delay in global wires and inefficient usage of them.

In this chapter NoC as a novel solution for connecting cores in SoC is explained. The chapter is organized as follow. Section 1.1 demonstrates the concepts and the importance of NoC. Section 1.2 addresses NoC's characteristics. Section 4 illustrates NoC's architecture. Technology trends related to NoC which makes NoC testing crucial and NoC testing are discussed in sections 5 and 6 respectively and last section conclude the chapter.

2.1 Concept and Importance of NoC

In today's designs, as the complexity of SoCs is growing rapidly, two major concerns of manufacturers are time to market and the cost of the design. Time to market is the time from designing a chip until it is ready to be sold in the market. The cost of each SoC can be computed by equation given below:

$$Cost-per-chip = \text{manufacturing cost} + \frac{\text{design cost}}{\text{number of sold chips}}$$

According to above equation, in order to decrease cost of SoC we can:

- **Decrease cost of manufacturing.** This is realted to technology which is being used for implementing the chip.
- **Increase number of chips sold.** This is possible by having more flexible designs like general purpose processors versus *Aplication Specific Integrated Circuit (ASIC)* and reusing them in multiple applications.
- **Decrease chip design cost.** It means we should design more efficiently by applying special trends as reusing blocks (IP cores) in design and similar trends that are listed below.

In order to reduce the chip design cost, we can use number of methods as:

- **Blocks reuse.** It means design should be modular.
- **Architecture reuse.** This means that the way different components are connected to each other should be reused. It necessitates employing platform-based designs.
- **Scalable architectures.** In scalable architectures number of blocks can be increased without changing the architecture.
- **Abstraction.** It is used in complex designs to hide the details of implementation and handling the complexity by distributing the design in different layers of abstraction. Each layer has some special responsibilities which is part of the design

process. But the layers do not interfere with each other and communicate only through interfaces in between them. Abstraction results in minimizing the number of elements on the chip. It also simplifies the interaction between the elements.

- **Compositional architectures.** This means different components(blocks) of the design do not interfere with each other. In this way if a new block is added to the design and we are sure about the correctness of its functionality, there is no need to test and re-verify the block; because we know different blocks do not affect each other. Therefore only testing the whole design (for assuring that blocks work well together) is enough.

For having cost effective SoC we can exploit aforementioned trends in both on-chip blocks (computation part of the chip) and on-chip connections (communication part of the chip).

In today's designs, in order to manage the complex SoCs, designers are trying to reuse existing cores. In this way by having pre-generated IP cores (provided by core providers) the SoC designers only have to deal with connecting the cores in an efficient way [2]. Conventional on-chip interconnects (such as Amba bus and Silicon backplane) can offer a limited amount of bandwidth. Moreover, they do not satisfy the performance (latency and bandwidth) requirements of today's complex on-chip communication [3]. More advanced solutions for interconnects are Multi-layer Amba busses and other similar solutions that can provide larger bandwidth for communicating between the cores. However they can not offer the required scalability for cost effective communication scheme [4][5]. For having a desirable on-chip interconnect in terms of cost, with all of the mentioned trends, SoC designers are trying to replace design specific global on-chip wiring with a general purpose on-chip interconnection network [4], referred to as NoC.

Open Systems Interconnection reference model (OSI)[6] is a standard (defined by Communication Networks Community) for communication between computers when they are networked. This standard partitions the communication problem in the networks into a number of smaller problems and group the problems. Each group of problems is solved in one section of OSI model, named as *layer*. In OSI model there are 7 layers (each layer and its functionality is explained in what follows). The so called layers are isolated from each other and can interact to each other only by well-defined interfaces. In NoC the same layered model (OSI model) is employed to solve the communication problem between the cores that are networked in SoC.

In Figure 2.1 only three layers of OSI model in NoC are depicted to show the way they interact to each other . As depicted in Figure 2.1, each layer may include one or more protocol functions operating on data units at different levels of abstraction like analog wave forms, bits (data word), packets and streams. As an example data unit in link layer is data word; while in network layer packets are units of data. Protocols are a set of control algorithms that are between identical pair of layers in the sender and the receiver, e.g. between two link layers or between two network layers. The protocols define how cores interact in order to satisfy both the communication and the performance requirements [5] [7].

Figure 2.1 also shows that each layer offers service to its upper layer; while it receives services from its lower layer, e.g. network layer gives service to transport layer and

receives service from link layer.

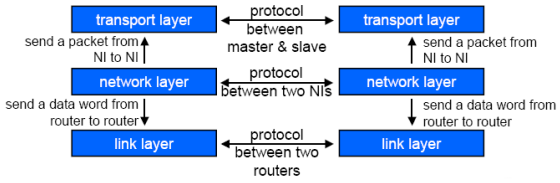


Figure 2.1: Layers and protocols between layers [18]

Detail functioning of each of the 7 layers is given below; we start from the lowest level of abstraction to the highest level:

- **Physical layer.** This layer is concerned about the lowest level details about the data transmission. These details define signal voltage, timing, bus width and pulse shape, and the synchronization of signals (because the IP cores may be in the different clock domains).
- **Data link layer.** This layer is responsible for reliable data transfer over the physical link. Therefore it may have features like error detection and error correction functions. Another key responsibility of this layer is arbitration.
- **Network layer.** This layer provides end to end connection between the routers by defining routing information, which can be statically or dynamically defined. However the topology of the routers is not evident yet.
- **Transport layer.** In this layer end-to-end connection between master and slave is established and maintained as well as the topology. Flow control, ensuring message ordering and packet segmentation and reassembly are other tasks of transport layer.
- **Session layer.** This layer adds state to end-to-end connections which are provided in transport layer.
- **Presentation layer.** It Concerns with the representation of the data in different forms of messages and converting them to each other.
- **Application layer.** This layer offers the highest level of abstraction of functioning of underlying layers to the system components. It performs communication between the components.

By partitioning, the communication problem into layers we change the problem to the more tractable and simpler problems. Therefore synthesis and validation of communication design will be also simplified. In this way abstraction for On-chip interconnect design is granted.

NoCs have delivered advantages beyond conventional on-chip interconnect and are suitable for high density SoC for future technology. By providing standard interfaces which are compatible with existing protocols, NoC facilitates reusing of IP blocks and increase modularity in SoC. Scalability of the system will also be possible by NoC scheme, since adding or removing cores through the well-defined standard interfaces becomes an

easy task. High capability of NoC for concurrent transmission of data, makes it possible to extend the SoC by adding cores to it.

Compositional architecture also becomes possible due to one of the most important characteristics of NoC which is *Quality of Service (QoS)*. QoS is the amount of percentage that a system can guarantee to give a specific service to its user. QoS is explained in more detail in NoC characteristics section. One type of QoS is *Guaranteed Service*. By having *Guaranteed Service* in communication part of SoC, it can be assured that the behavior of computational components (IP cores) when they are communicating to each other is the same as the time they are individual (not connected to each other). Hence by connecting an IP core to other cores there is no need for re-verification of the whole SoC. As a consequence the total cost and the time-to-market of the SoC design is decreased.

2.2 NoC Characteristics

Since NoC glues the IP blocks on SoC, it is seen like an IP block. Thus, in order to have a cost-effective chip, NoC should have similar properties as the IP blocks. Moreover, NoCs have some of the characteristics of off-chip networks. By using NoC, in addition to making SoC modular, on-chip communication itself also becomes modular due to the fact that communication functions in different layers can be reused in different on-chip networks. Another advantage of using OSI reference model in NoC is that it is scalable. Depending on the needs of system's connection, necessary layers can be added or unnecessary layers can be removed. If the connection does not need certain features, the corresponding layer to that feature can be eliminated. On the other hand, on-chip network differs in some characteristics from off-chip networks. These characteristics are specific to NoC and are explained below [2]:

- **Reliability and Quality of Service.** *Quality of Service (QoS)* is defined as "The degree of satisfaction of the user of the system". When a system gives a specific service to a user, QoS defines the percentage that the user can be sure about receiving the expected service. QoS has different levels of providing the wanted service. Levels of QoS are:
  - **Guaranteed Service (GS)**: In GS or *GT* services, hundred percent of times the required service by the application is granted by the system.
  - **Best Effort (BE)**: The system may or may not provide the required service.
  - **Statistical service**: The system provides the service a specific percentage of times.

  Considering NoC as a system, transferring packets in a limited time is the service that NoC provides to communicate cores on SoC. Since SoCs are used mostly in embedded systems that are usually real-time or safety-critical, QoS is essential in the SoC designs. For example in NoC, the connections which require QoS, are delivered before a maximum amount of latency. Another instance of service is providing a required amount of bandwidth for a connection. Connection is the set of links from source to destination that a packet travels through them.

The most important level of QoS is GS, since it helps the system to be predictable and robust. In NoC, GS for mentioned services (latency and bandwidth) can be acheived by *contention free routing* (with or without TDM) or by *packet switching* accompanied by proper arbitration scheme. If packet switching routing is exploited GS deeply depends on arbitration schemes used. Contention free routing and packet switching are routing schemes that will be explained in detail in the NoC architecture section. As an example Ethereal [16] is an NoC which uses contention free routing for providing GS for latency or bandwidth. However, MANGO NoC uses packet switching with a specific arbitration scheme to provide latency and bandwidth [17].

- **Low area overhead and power consumption.** Chip designers are trying to make smaller and more power-efficient designs to be used in different applications including portable devices. Therefore area overhead and power consumption are major concerns in on-chip designs. In on-chip networks conversely to off-chip networks, designers should be careful with number and size of buffers that are used. However, as a result of small buffers, the probability of buffer overflow increases. For solving buffer overflow problems, different schemes of flow control can be emplyed. But these schemes should not have a large overhead in terms of power and area. Furthermore, NoCs require fast and simple arbitration schemes to satisfy the latency-constraint applications. The arbitration schemes should also be efficient in terms of area and power.
- **High performance.** Besides necessity of decreasing chip area and power consumption, on-chip schemes should be much more faster than off-chip counterparts. Performance in NoC is defined as the ability of NoC to provide low latency and enough bandwidth (throughput) for the data transmission. High performance data transmission in NoC is acheived by QoS features.
- **Signal integrity.** Opposite to inter-computer networks that have long transmission lines, NoC does not contain long wires compared to its off-chip counterpart. Therefore crosstalk effects that are related to long wires significantly decrease in NoC. Another difference is that in NoC the number of wires that connect routers are much more than inter-computer networks, since they are short and relatively cheaper. Therefore, it is not crucial to be provident in using wires.
- **Predictability.** Another important feature of NoC in comparison with dedicated wires in buses and switches is having predictable electrical parameters. This feature is due to its regular structure. Having regular structure, in turn enables high performance circuits and improves duty factor of wires [4]. Increasing duty factor of wires, means making more use of wires and decreasing the amount of time that they are idle.
- **Work conservatism.** It means that sources of system should not be wasted and the user should get the most use of them. For example the amount of time that lines are idle should be decreased, or the bandwidth should not be wasted (When a connection does not have any packet to be sent it should not occupy a portion of bandwidth). Special trends is applied in NoCs such as MANGO [17] and Ethereal [16] to be work conservative.

Special characteristic of NoC (which is similar also in off-chip networks) compared to buses, is *packet based communication*. It means that data is sent in standard packets. Structure of the packets is characterized by each NoC. Messages that come from cores to transfer through NoC will be segmented to segments as packets. Each packet consists of a number of flits (which is the unit of data transfer in NoC) and has a header flit which contains the routing information of the packet.

2.3 NoC Architecture

NoC architecture is composed of routers, *Network Interfaces (NI)* and links. Links connect the routers together and also the routers to the NIs. Network interfaces are placed between a router and an IP core and is used as a mediator between IP cores and the interconnect network. Routers are responsible for routing data units of transmission from source to destination. Figure 2.2 shows the general architecture of NoC.

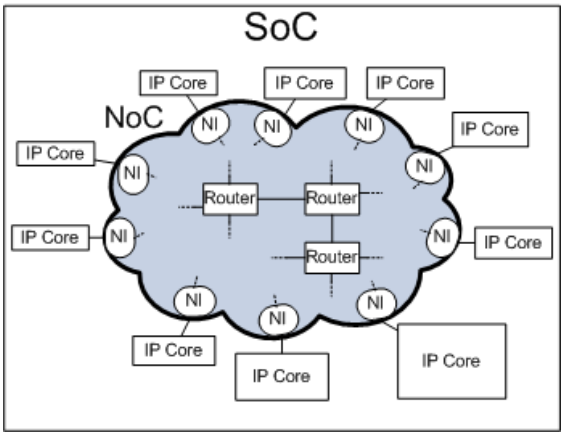


Figure 2.2: General architecture of NoC

Below there is explanation about each of the network components, but first the format of data being transferred in NoC (packet) is explained.

2.3.1 Packet format in NoC

When two IP cores want to communicate (master and slave pair), there will be request and response transactions between them. Each transaction between master IP core and slave IP core contains a message.

Message that is being transfered by the NoC should have a special format. This format is the *packet* format. Each packet is composed of smaller data transfer units named as *flits*. Flits are the smallest possible units that can be transferred between hops in the path in NoC [30]. Flits, are composed of actual physical words named as *phit*. Each *phit* is the actual physical data units that construct the packets. When a packet is being transferred, it will be transferred *phit* by *phit*.

The first flit of the packet is the header flit.The header flit has the control information like the path to destination, the type of packet (BE or GS).

2.3.2 Links in NoC

Links in NoC are used to transmit packets or flits between routers. The major problem with links is the long wires which are becoming main source of delay with progressing technology; as a consequence it causes increasing link access time. For resolving such a problem, NoC pipelines long wires in interconnects by partitioning the wires into number of segments depending on communication requirements [9]. In the pipelined links the longest delay in the pipeline segments determines the clock latency. The total delay of the link would be the number of pipeline segments multiplied by clock latency. In this way, the number of transmitted packets (throughput) via a link in a specified time (equal to clock cycle time) will be increased. However total delay that each packet or flit is facing with is not decreased but maybe even increased. This is due to the fact that the latency of each segment will be the largest latency among all of pipeline segments.

2.3.3 Routers

Routers or Switches consist of a combinational circuit for routing packets (refered to as Routing Logic Block (RLB)) and input/output ports. RLB connects the input ports to the proper output ports. Depending on the routing scheme, routers may or may not have buffers. Figure 2.3 depicts block diagram of a router containing buffers.

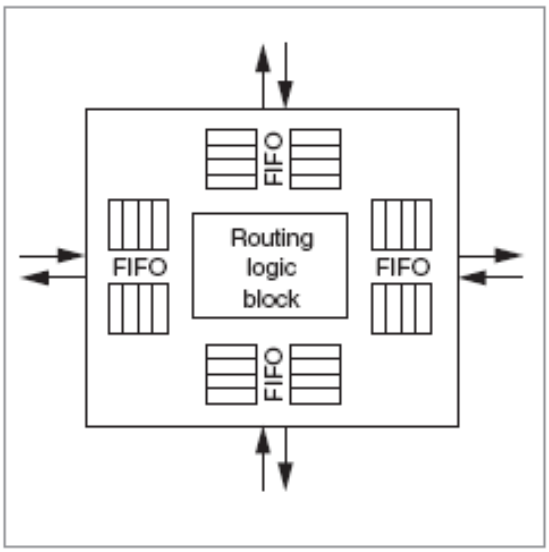


Figure 2.3: Router architecture containing buffers [10]

Routing schemes are:

- **Contention free routing.** This routing scheme is based on buffer-less routing (There is no buffer inside the router architecture). It is accomplished by either

circuit switching or by time division circuit switching. In circuit switching, the links are reserved for a special connection from the source to the destination. In time division circuit switching, the time to access the shared link can be divided into slots between different connections and in each time slot the link is reserved for a special connection.

- **Packet switching.** In packet switching, since links are not reserved for connections and we do not know the arrival time of packets (or flits), there would exist a contention for accessing the shared resources (link or buffers). For overcoming the problem of contention there is a need for arbitration schemes between contending connections. The arbitration scheme defines which connection is eligible to access to the link and which connection should wait to access the link. Therefore, there should exist a buffering scheme for storing packets (or flits) of connections which loose in the arbitration and should wait. Buffers can be placed at the input ports (input buffering) or at the output ports.

There can exist different types of buffering schemes in the router. Input buffering is cheaper; however it suffers from head-of-line blocking [18] which causes performance loss. Output buffering is more expensive but provides better performance. In some routers each port consists of a number of virtual channels. For output buffering, each virtual channel has its own buffers; this leads to increasing cost in output buffering. It is also possible to have both the input and the output buffering.

Depending on the communication requirements different types of flow control can be exploited in the packet switching routing. The type of flow control determines the unit of data transmission in NoC [9]. Therefore the amount of required buffer in the router will be also determined. They explained below:

- ***Store and Forward (SAF).*** In this method of flow control packets are the units of data transmission between the routers. A packet is sent to the next router when there is enough space in the next router to accept an entire packet. The latency of data unit transmission is equal to the packet transmission and buffer sizes should be at least equal to the packet size [9].
- ***Virtual Cut Through (VCT).*** Although in VCT data transmission starts when ever there is enough space in the buffers of the next router for complete packet size, unit of data transmission is flit. This causes to have less latency (equal to transmission of flit) than SAF; while having the same size of buffers as SAF [9].
- **Wormhole.** Here flits are units of data transmission and condition for allowing the data transmission is that there is enough space in the next storing buffer for one flit. Thus the transmission latency is low as well as the size of buffers. However if there is not enough space for one of the flits of the packet, the following flits also can not transmit and stay in their current buffers. This causes the packet to spread in different routers. As a consequence wormhole routing is more sensitive to deadlocks [9].
- ***Virtual Channel (VC).*** VC is like wormhole; however here each physical link consists of a number of independently buffered virtual channels. Therefore if one

of the virtual channels is blocked by a flit of a packet, the other packets can be transmitted through other virtual channels and possibility of deadlock is decreased. But it is more expensive than other flow control methods.

2.3.4 Network interface (NI)

Network interfaces are between computational cores (IP cores) and communication fabric (NoC). Main responsibility of them is to decouple communication part of SoC from computational cores on the chip. This helps designers to optimize communication and computation parts independently [9]. In fact NI implements features in transport layer like end-to-end flow control (which matches sender and receiver rates) and error control. Features done for decoupling communication and computation are:

- Converting messages that are produced by core transactions to packets that are data units in NoC and vice versa.
- Protocol conversion.
- Clock domain crossing.

NI can have a layered architecture like the one in Ethereal NoC [15]; see Figure 2.4. Layered NI is composed of a kernal and a shell, which makes the interface more modular and distributes responsibilities between two segments.

The first segment; *kernel* is between the shell and the routers and is responsible for:

- **Packetizing and de-packetizing.** The conversion of messages to packets is packetizing. On the other hand, the conversion of packets to messages is de-packetizing.
- **Packet insertion to NoC.** NI kernel send packets to the first router of the path from source to destination.
- **Qos checking.** NI kernel assures that each connection will receive the QoS that it supposed to receive.
- **End-to-end flow control.** NI kernel, is responsible for checking if there is enough space in the remote NI. If there is enough space, then it will send the packet (flit).

  The second segment; *shell* is placed between the kernel and the standard ports (that are between IP cores and NIs). The shell is responsible for the items listed below:

  - **Buffering.**
  - **Conversion between clock domains.**
  - **Conversion between data widths.**
  - **Supporting different types of connections.** There are more types of connections rather than peer-to-peer connections (like narrowcast and multicast connections).

  In the layered version of NI, the shell is implementing the functions in the transport layer and the kernel is implementing the functions in the network layer.

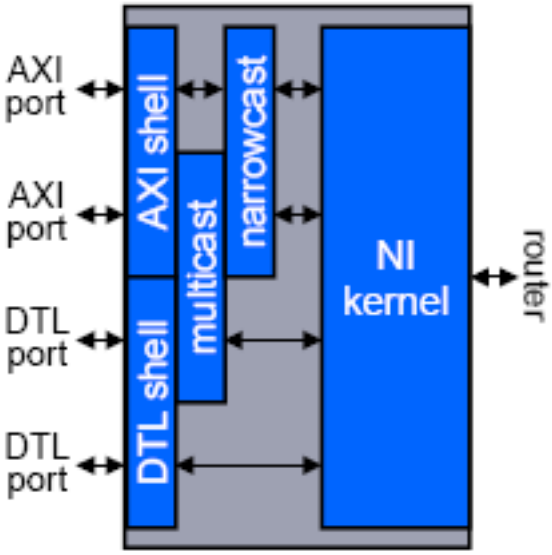


Figure 2.4: Layered architecture of network interface [15]

2.3.5 Buffer management in NoC

If the routing in NoC is buffered routing, before sending data to the routers, there is a need to control if there is enough space in the buffers of the routers or not. This is called *link-level* flow control. In link-level flow control, each router should be assured that there is enough space in the next router of the path and then the data will be sent to the next router.

Between the sender NI and the receiver NI, there should exist *end-to-end* flow control. In end-to-end flow control the sender NI controls the amount of available space in the receiver NI. If there is enough space in the receiver NI to receive one *unit of data*[1] the sender NI will send the data.

2.3.6 Topology

The way routers are connected to each other is defined by network's topology. Network' s topology is an important issue in NoC, since it has significant effect on efficiency of on-chip resource (link) usage. The NoC topology also affects the control of physical issues (electrical parameters). In regular topologies (like mesh) there is a better control on electrical parameters and noise sources (like crosstalk) but it may suffer from under-utilization of links and localized congestion. However in irregular topologies there are less control on physical issues, instead it is more efficient in resource usage [9]. Different topologies for NoCs trade off between various characteristics of NoC such as performance, reliability (QoS), work conservatism, power dissipation and silicon area [10]. Topologies for NoC can be regular like Mesh [11], Butterfly fat-tree [12] and Octagon [13] as shown

[1]The unit of data is determined by the type of flow control, e.g SAF, wormhole, VCT or VC.

in Figure 2.5a and 2.5b and 2.5c, or it can be a irregular application-specific NoC [14], depicted in Figure 2.5d .

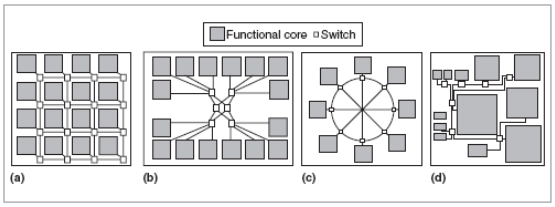


Figure 2.5: NoC regular and irregular topologies [10]: Mesh-based(a), Butterfly fat-tree(b), Octagon(c), Irregular application-specific(d)

2.4 Technology trends related to NoC

As the technology scales down, ICs are getting smaller and smaller, chips are becoming faster, cheaper and more power-efficient. On the other hand technology downsizing brings a set of new problems related to reliability and error-free functioning of on-chip elements. Recently, SoC designers by increasing the operating frequency and the transistor density on the chips, are able to build faster and smaller chips. However with increasing the frequency and the density on the chips, more energy will be consumed which leads to undesirable increasing of on-chip heat. In order to compensate for this incident they reduced the on-chip voltage. However it should be noticed that another main reason of power dissipation in SoCs is related to leakage. For fixing this problem *Dynamic Power Management* techniques should be employed [18].

All of these new trends (decreasing sizes, decreasing on-chip voltage and increasing frequency) used in SoCs makes the design more sensitive to the internal and the external noise resources (which are the reasons for malfunctioning of the system).

Sources of errors in SoC' s cores and its interconnects can have different roots from the process technology to the design style [18]. They are discussed next.

- **Process variation.** The designers design the chips based on known parameters like length of the channels or their thickness. However, after manufacturing the design, these parameters may have slightly different values than it was expected. By moving toward nano-scales, these small variations can have more effect on design, like producing delay faults.
- **Process technology resources.** Process variations and defects during manufacturing can cause crosstalk defects in SoC interconnects. Signal integrity can cause significant defects in proper functioning of SoC's interconnects[19] [20]. Signal integrity is mainly due to cross-coupling capacitance and mutual inductance. Cross-coupling capacitance and mutual inductance is because of interaction between two parallel interconnect lines. Figure 2.6 shows circuit model of two adjacent interconnects, their distributed values of each line (capacitances, resistances

and inductances), the cross-coupled capacitance between two lines and mutual inductance between them.

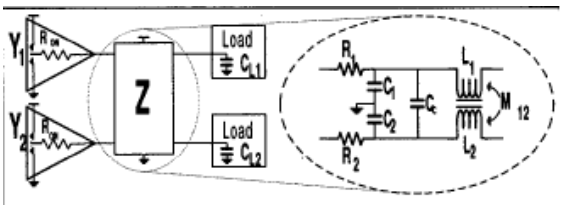


Figure 2.6: Circuit model of adjacent interconnects [20]

In sizes of sub-micron processes, resistance and capacitance of line, load capacitances (C_L) and driver resistance (R_on) are the factors that determine the behavior of circuit. However in deep sub-micron sizes, this is the cross-coupled capacitance (C_c), line inductances and Mutual inductance (M_12) which defines the behavior of circuit and is responsible for generated noises [20]. Increasing these factors that are responsible for crosstalk and signal integrity is due to a number of physical origins like decreasing space between lines, increasing height to width ratio of each interconnect and increasing density. In [21] the authors, by comparing CMOS 0.7_mm and CMOS 0.18_mm technologies, show how scaling down technology is affecting density in a chip and therefore signal integrity. In CMOS 0.7_mm there are two metal layers and the silicon surface is 40*60 mm; while in 0.18_mm CMOS number of metal layers has grown to six layers and the silicon surface has diminished to 10*5 mm. Hence, it is obvious that by downsizing technology the density on the chips increases.

As it can be observed, there are several conditions leading to crosstalk. Therefore for facilitating crosstalk analyzing, *Maximal Aggressor Fault (MAF)* model is used to model crosstalk [19] [20] [22]. In MAF model, we consider all lines to be aggressor which are affecting only one line which is victim. Cross-coupled capacitance can cause three basic signal abnormalities which are depicted in Figure 2.7. When all aggressors are switching from logic high to logic low or vice versa and the victim line is stable, coupling capacitance will transfer energy to the stable line and an unwanted glitch (positive or negative) will be produced on the victim line. Moreover when victim is switching logic (high to low or in reverse) and all aggressors are changing in opposite direction, they affect the victim signal by increasing its transition time, leading to falling delay or rising delay in the victim line [19] [20]. Crosstalk effects can also cause unwanted speed up in wires, that is when all agressors and the vistim are changing logic in the same direction.

Besides when inductance is combined by other elements in circuit, it produces damped voltage oscillation which can cause long-range noise. However, the effect of inductance noise can be approximated by effect of cross-coupling capacitance noise in many cases [19].

We can reduce crosstalk effects by analyzing responsible factors for crosstalk dur-

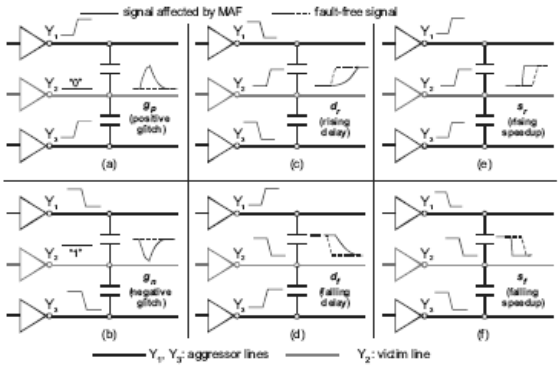


Figure 2.7: Crosstalk errors in MAF model [19]

ing design; however process variations and defects that may happen during manufacturing may also affect cross-coupling capacitance which is the main source of crosstalk and delay defects [19] [20]. As an example of manufacturing defects in [20], they analyze the way that a floating net can increase cross-coupling capacitance between two adjacent lines. In Figure 2.8 if Y3 becomes floating in manufacturing phase, Y1/Y3 capacitance and Y2/Y3 capacitances become coupled and increase cross-coupling capacitance between Y1 and Y2.

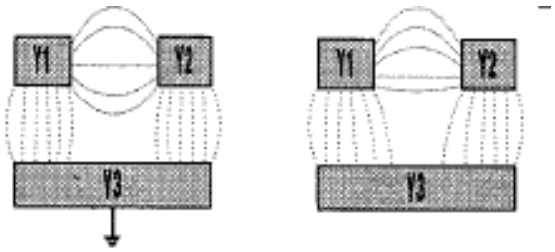


Figure 2.8: Effect of floating conductor on cross-coupled capacitance [20]

- **Environmental resources.** Instances of errors that can be stimulated by the environmental resources are Electromagnetic Interference (EMI). EMI takes place due to external sources of noise. With reduction of the voltage swing for compensating power consumption EMI can cause more defects on chips.

  Soft errors are the most popular effects of radiation in the ground applications. When a soft error occurs it causes a data bit corruption until the time that it is overwritten again with a new data [23]. Soft errors are result of environmental resources like radiation. Main radiation reasons that cause soft errors are alpha particles, high energy neutrons and also low energy cosmic neutrons. Each of them is consequence of some radioactive isotopes (that exist in packaging material) or interaction of cosmic ray with earth atmosphere [23]. Alpha particles are electrically

charged and create a track of electron hole pair when they pass through the IC. If a sensitive node (like reverse-biased junctions) is close to this track, it collects the generated charge that causes a transient current pulse on the sensitive node [23,24]. On the other hand, neutrons are electrically neutral and can not create an electron hole track directly. But they interact with material elements that exist in substrate and produce electrically charged secondary particles that may cause track of electron hole on the IC [24].

The transient current is like a pulse, depending on which type of node it is produced, it can cause two types of soft errors. First, if the node is a latch, flip-flop, register or any kind of memory cell (like buffers), the undesirable pulse may turn over the content of the cell. This causes an error called Single-Event Upset (SEU) [23,24]. Figure 2.9 depicts the effect of SEU in a memory bit. Second, if the node is a combinational logic node, the current pulse converts to a voltage pulse which is called Single-Event Transient (SET). The unwanted voltage pulse on logic node can propagate through one or more path, if the inputs of the gates composing the path have non-controlling values (controlling value for NAND gate is a 0 and for NOR gate is a 1, that blocks the propagation of voltage pulse) [24]. The SET can also affect the memory cells if the propagated pulse reaches to a latch or flip-flop and it overlaps with a clock event. It reverses state of the latch or the flip-flop. Another issue that may prevent pulse propagation is when the pulse width is smaller than the gate transition time.

SEU is the main reason of data corruption in memory cells because as mentioned above, occurrence of SET depends on a number of conditions. However with increasing operational frequency SET is also becoming a threat for data corruption in memory cells.

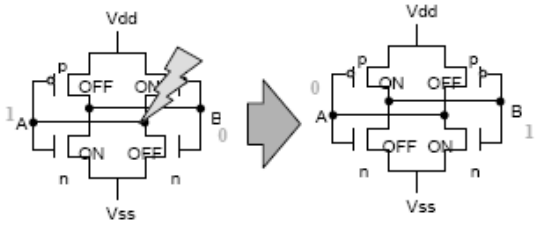


Figure 2.9: Illustration of the SEU effect in a memory cell [22]

- **Operation mode reasons.** Considering design requirements such as the high speed and low power circuits, designers may increase the operating frequency of chip or decrease operational voltage of chip. Both can cause more defects on the chip.
- **Design style reasons.** As mentioned in first section, employing abstraction in layered approach of on-chip communication helps designers to deal with increasing complexity of SoCs and especially SoC 's complex interconnects. But, by using abstraction and hiding implementation details of the design, designers can no longer

have control on such details. This leads to increasing *variability* of electrical parameters [18]. In other words by abstracting details of lower levels of design (such as physical and electrical details in physical layer), there is no control on these details. Different events such as temperature variations may affect lower level detail's functionality and make them variable. Increasing variability in lower level details makes the system non-deterministic. In non-deterministic systems, we can not expect the events to be the same as the way that we predict them to be Therefore, we should test for defects that are possible to happen due to non-determinism of system.

Errors originating from different sources are independent of each other and can take place in a circuit at the same time. Thus, they can double the effect of errors in a design or it is also possible to compensate the effect of each other. But it is a rare case that they neutralize each other. As an instance in NoC, different elements are victims of different sources of errors. As an example, wires between routers are subjected to crosstalk and signal integrity errors, while buffers in routers are involved with SEU errors.

In [22], it is claimed that crosstalk and radiation are essential reasons for errors and if they occur together at the same time in NoC they can cause crucial problems. For proving their claim they have developed and simulated a fault injection system based on crosstalk and SEU fault models to see the resulted fault effect. The fault injection system is composed of a MAF model and SEU fault model. The MAF model produces glitches or delay faults in a NoC link that connects two routers. SEU fault model forms an undesirable bit-flip in bits of buffers placed in routers. They show that by different combinations of these two errors occurring in different times and different parts, different fault effects can happen. Some of the observed fault effects are wrong packet routing, packet starvation from routing, packets being overwritten, changing packet payload or framing signals and router crash (which is a permanent defect and needs hardware reset).

2.5 NoC and Testing

Three steps to test the whole SoC are [10]:

1. **Testing NoC infrastructure.** Testing NoC infrastructure consists of testing elements of NoC; links, routers and NIs.

2. **Testing IP cores.** This includes testing functional blocks (Processors or Modules like MPEG or PCI) and storage blocks (Memory). Sometimes, the NI that is attached to an IP core, can be tested with the IP core at the same time.

3. **Testing the whole integrated circuit.** In this step the interaction between IP cores and the interconnect will be tested. It is worthy to mention that the NI blocks can be tested either with testing the functional blocks or with testing the NoC. An interesting issue about NoC is that it can be used for testing IP cores existing on a SoC through the NoC and testing the NoC infrastructure itself. However, in order to use NoC as a tool for testing other cores on the chip we should first test the NoC and be assured that each of the components in NoC are error free [2].

As it is explained earlier, today's chip design is becoming communication-centric. This means a faulty NoC leads to destroy the whole SoC and if NoC fails in testing there is no need to test the rest of system. Therefore, by testing the NoC first, the test time and test cost can be reduced[2]. Moreover NoC may be used in testing the IP cores. Thus the order that three steps of testing the SoC should take place is: testing NoC, testing computational on-chip cores and at last testing the whole integrated circuit. More information about each phase of test is given below.

2.5.1 Testing NoC infrastructure

Although NoC is a new feature given as solution for today's on-chip problems, in order to practically be useful, problems related to its testing and verification should be solved .On behalf of structural testing of NoC three main elements should be tested; routers (switches), NIs and links.

The order of testing these elements is also important; links have to be tested prior to switches since inter-switch links are responsible for transferring data between switches. Therefore testing links is the first step for insuring correct functionality of NoC. Moreover in some design methodologies the NoC infrastructure is reused for testing NoC's routers. It is reused for transferring test data to the routers under test. In such cases, links are the basic elements for transferring test data, therefore we should first guarantee proper working of these wires and then use them as test media.

One important characteristic of NoC testing is that due to regular structure of NoC that uses identical elements (switches and links), only three types of test data is necessary; test data to test all switch blocks, test data to test NIs and another test data is used to test all inter-router links. If NoC infrastructure is reused to transport test data, we can take advantage of parallel data transmission in NoC. Therefore one type of test stimuli can be broadcast in parallel to a number of identical elements (e.g. a number of routers). this feature leads to reduction of test time.

It can be seen that in routers we need to test two parts; The combinational logic part used for routing data (RLB) and the buffers. Although testing combinational logic part is not a complicated issue and can be done by traditional methods of testing like scan and BIST [10], testing of buffers in switches is not a straightforward task. The main reason for difficulty in buffer testing in NoC is that they are small and distributed all over the chip. Therefore, if we want to employ the traditional BIST method (which is one BIST architecture per buffer), it leads to a large area overhead which is not a desirable fact for today's design [10]. Thus new features like embedded distributed BIST structures are needed for testing buffers.

Considering the above mentioned facts about NoC testing, one of the major challenges of NoC testing is to prepare proper test patterns for stimulating faults. Each test stimuli should cover faults that are possible to happen to the element under test. For this reason proper fault models should be exploited. for example gener-

ated test data for wires should cover crosstalk faults that are main source of error in wires and use MAF model for this purpose.

One of the most necessary test patterns that should be developed for NoC links and NIs, is timing tests. This is due to two main reasons [2]. First, recently NoC is being used in *Globally Synchronous Locally Asynchronous (GALS)* systems. In GALS systems it is allowed to integrate independent cores that operate at different frequencies. Therefore, different clock domains should be synchronized asynchronously [8]. Thus all NoC components (such as NIs) are responsible for synchronization of different clock domains together. Second, since NoC is spread all over the chip, in large scale SoCs, there are relatively long wires in scales of on-chip wires. Therefore they are more in danger of delay faults due to crosstalk effects.

2.5.2 Testing IP cores in SoC

The most important feature of SoC is to make it possible core reuse which leads to reducing time to market. However if this feature only limits to core design, core testing becomes a bottleneck. Hence core tests should also be reused in order to increase efficiency and reduce time to market. By development of SoC design, core providers are responsible to design core and give them to SoC integrators. Since these cores are given as encrypted IP cores, the system integrator does not have any knowledge about the detailed implementation of the cores. This issue, makes it impossible for system integrators to develop proper test methodologies to test the corresponding cores. Consequently, core providers took the responsibility for providing a pre-generated set of test patterns for each core that they give to the system integrator. In this way the system integrators only have to apply the pre-generated test patterns to the cores and then observe the responses.

Since the cores on SoC are deeply embedded, accessing to their terminals from I/O pins and applying test data to their terminals and observing their output is not a straightforward task. For solving this problem, there should be a special hardware for transferring test data from I/O pins of chip to core terminals. This hardware is called Test Access Mechanism (TAM).

Other necessary issues for testing on-chip embedded cores are a mechanism for isolating the core under test from its surrounding cores (because during testing a core it may damage its surrounding cores). This mechanism should also switch the core-mode between functional and test modes. Therefore, the test functions such as *Built-In-Self-Test (BIST)* or scan chain functions can be activated during test mode and deactivated during functional mode [25]. This two last mechanisms can be realized by a dedicated on-chip hardware which surrounds around each core, named as wrapper.

Since different types of cores can exist on SoC, there should be a standard between core providers for test data generation. *Test Technology Technical Council (TTTC)* of IEEE has established this standard which is IEEE P1500 [2][25]. IEEE P1500

uniforms the generated test data by the core providers and also facilitates testing embedded cores. It has two major parts;

- ***Core Test language (CTL).***: This is for standardizing the transferring of test data.
- **Scalable wrapper architecture.** This architecture standardizes the wrapper and the wrapper port to TAM.

Figure 2.10 depicts the structure of applying test data to the the embedded cores in IEEE P1500.

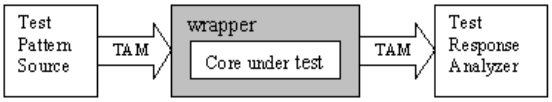


Figure 2.10: Schematic view of applying test to embedded cores in IEEE P1500 [2]

TAM can be realized by having a dedicated hardware for it or the on-chip interconnect can be re-used in the role of a TAM. NoC which is the new paradigm of on-chip interconnection can also be re-used for this task [26]. By reusing NoC as TAM, test data should be formed in packets to transfer through the NoC (as mentioned earlier NoC is packet-based). The key advantage of this type of TAM is concurrent transferring of test data to different cores under test. This leads to reduction of test time. It is also advantageous in terms of cost, since there is no need for extra hardware for dedicated TAM. On the other hand concurrent transmission of test data has a downside which is an excessive power dissipation [10]. Therefore, proper utilization of test parallelization should be done by trading off between power consumption and test time.

2.5.3 Testing the whole integrated circuit

Testing on-chip communication structure and on-chip computation cores separately is not enough to be sure about the correct functioning of whole system. Thus, interaction between different computation cores and on-chip interconnect should also be functionally tested. Major issues to be tested for this part is I/O functionality of each processing core and functionality of routers [10].

2.6 Categorizing methods of embedded core testing

Generally, we can categories methods of testing by two aspects. First, is that testing can be functional or structural. Second, is that testing can be done offline versus online. These aspects of testing are explained below.

2.6.1 Functional versus Structural

In order to give define what is functional and structural test, first some definitions should be explained; Defect, Fault and Fault Model.

Definition 1. Defect is the difference that happens between the design and the actual implementation of the design [27]. It means that designers expect the implementation of the design to work exactly like the actual design but due to some reasons the implementation may be act different. For example, it may have short between links that is not wanted or other kinds of defects.

Definition 2. Fault is the effect of the defect on the system. It can be defined as faulty output signal of the defective system [27].

Definition 3. Fault models are abstract representations of defects and can be in different levels of abstraction. Figure 2.11 shows different levels of abstraction that a fault model can be presented which are fault models at layout level, electrical level, logical level, functional level and behavioral level.

The number of defects that may happen is too many they may cause number of different faults that are not easily analyzable. Therefore, for facilitating the analysis of faults and finding the defects (which are the source of faults), fault models are used.

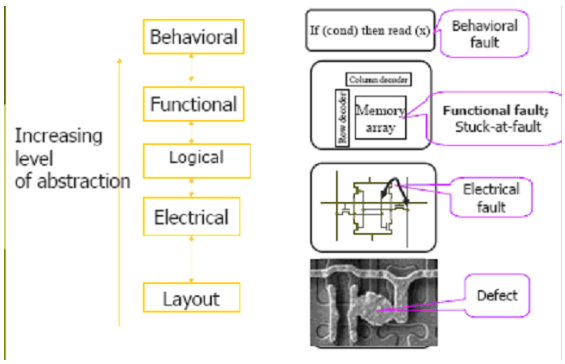


Figure 2.11: Fault models in different levels of abstraction [27]

The simplest example of fault models is the Stuck-at fault model which represents short or open defects. Other important fault models are MAF (which represents crosstalk defects), SEU and STU . Each of the MAF, SEU and STU fault models is being used in special parts of NoC, depending on what kind of errors can occur in each part. We will explain more details about each fault model in the corresponding section of NoC testing.

Depending on the level of abstraction of the fault model which is being used by the testing method, we can classify the testing method to structural and functional testing.

2.6.1.1 Functional testing

Functional testing is a comprehensive testing which tests if the *Device-Under-Test (DUT)* can behave as it is expected to behave. It can be done in two different ways:

(a) **Applying functional test vectors.** By applying functional vectors to the DUT and comparing the response outputs to the expected output. As an example, in a 4-input AND gate, all combinations of the inputs (which will be $(2^4 = 16)$) should be applied at the inputs and the response of the AND gate to all of these combinations has to be checked with the expected answer. For this kind of functional test, there should be an *Automatic Test Equipment (ATE)* to produce a large number of functional test vectors. Another important characteristics of this method is that the quality of test (fault coverage) is dependent to the number of functional test vectors applied to the DUT. As an example, for a DUT that is not a critical part of a system the required test fault coverage is smaller than a DUT which has a essential role in the system. Therefore the number of functional test vectors need to test a DUT and thus the cost of functional testing of a non-critical device in system is less than a critical device in the system.

(b) **Using hardware redundancy.** Functional testing can also be done by using hardware redundancy in the design. In this method, the results of the DUT and the redundant module that is added for test reason will be compared. This method of functional testing does not need costs related to ATE machine to produce functional test vectors for DUT, but it has costs related to hardware redundancy. This cost can be reduced by using more elaborate techniques in redundancy like information redundancy. For information redundancy different kinds of error-detecting coding can be used, e.g. arithmetic coding, parity coding and etc. Error-detecting codes also have hardware redundancy which is due to the addition of extra bits for storing check bits. However, the hardware overhead related to error-detecting codes is in the extent of some extra bits and is not much. This kind of functional test is in the range of online testing methods and fault tolerance systems.

2.6.1.2 Structural testing.

Structural testing is testing the structure of the device-under-test by using lower levels of fault models. For example, structural testing of the same AND gate (mentioned above) tests all the input lines and the output line for stuck-at faults. Therefore the total number of test patterns that should be applied to the gate is 10 test patterns. If the tests were applied to an AND gate with larger number of inputs or more complex logic, the difference between the number of patterns to be applied in functional testing and structural testing was significant.

Structural testing itself can be divided into two classes [28]; static structural test and at-speed structural test. Static structural testing addresses only static faults

like stuck-at faults. However with downscaling of technology dynamic faults such as delay faults are becoming more and more important. Dynamic faults can be detected only through at-speed structural tests.

The simplest way to do structural test is to use *Boundary Scan (BS)* test. BS enables static structural test. By using BS test, scan chains provide a delivery mechanism for transferring test data from boundary of the chip to devices under test. The fault coverage of structural test depends on the test pattern that is applied and the ability of these test patterns to stimulate possible errors in the device-under-test. In at-speed structural test, two-vector test patterns are needed to apply transition to the device-under-test and then capture the responses to the transitions in a certain amount of time [28]. For enabling at-speed structural test, we should extend the BS test with a BIST method to apply these two-vector patterns. By employing BIST method accompanied with scan design, the need to ATE to provide proper test pattern at clock rate is eliminated [29].

2.6.1.3 Comparison between functional and structural test

Functional testing is the superset of both static and at-speed structural testing [28]. This means, wherever structural testing is used functional testing can also be used (but it should be assured that functional testing that uses test vectors for testing has enough memory for storing patterns.). Therefore, Functional testing is more comprehensive and covers both static and dynamic faults.

On the other hand it should be noticed that functional testing is expensive and time consuming. To achieve a high level of fault coverage, the cost (related to producing test vectors) and test time of structural test is less than functional test. In today's at-speed functional test, which uses test vectors, ATE cost is a major concern that may make the test of a device more expensive than its design [29]. Besides, since functional test vectors are extracted from system simulation, functional test increases the time-to-market [29]. Another problem of functional test is that, although it has high fault coverage, but it can only detect the fault and it can not provide any information on the location that fault that has been occurred. This means it can not diagnose the fault.

It is efficient to try to limit the use of functional test and wherever is possible to use structural test. In [29] it is shown that number of faults can be detected only by special functional tests, therefore for detection of those there is no remedy rather than functional test. When replacing structural testing instead of functional testing, it is extremely important that the structural test has enough fault coverage.

2.6.2 Online versus Offline testing

Offline testing refers to testing the chips when they are not in the application yet (e.g. they are still on the wafer or they are packaged but not in a system.)

Online testing is the test of chips when they are in the application. We can further divide the online testing into two sub-classes. First sub-class is when the device

that is in application is being tested, we stop normal operation of the device and convert to test mode. Hence, in this sub-class, the device is either in operational mode which does its normal job or is in test mode which is being exclusively tested. Second sub-class is when the device is being tested it is in the application at the same time. In the second sub-class, we do not switch between operational mode and test mode. Therefore, testing the device is done concurrently with the time that the device is doing its normal operation. In the latter sub-class, the device does not stop its normal operation for being tested, therefore the test time is expected to decrease compared to the former sub-class.

In today's technology, designers are trying to design critical parts of the design in a way that they are more robust against faults. In other words the designs are becoming fault tolerant. For fault tolerant devices a solution that can be provided is having back up. If an error happened in some critical parts the back up can operate and prevents the whole system to fail. Drawback of this conventional solution is the increasing cost of the system. Another solution for making a system fault tolerant is that while system is operating, we detect and correct the errors. It should be noticed that for this solution being cost-effective, we should detect the exact location of failure. Thus, we spend extra effort to repair only the defected part.

Due to the above mentioned about the functional and the structural test, one can conclude that structural testing is more promising. Because it can define exactly which part of device-under-test is faulty. On the other hand, in order to not degrade the performance of the system (increasing latency), online testing is preferable than offline testing, since it can reduce test time by testing the device-under-test concurrently with its normal operational mode.

Therefore fault tolerant schemes are needed for critical parts of a system. It is aimed in the future to move toward online structural testing. In today' s systems NoC is one of the vital part of SoC and needs to have fault tolerant features.

2.7 summary

Due to Moore' s law, number of transistors per chip are increasing exponentially and at the same time their size is decreasing, meaning that computation is becoming cheaper and cheaper; however this is not the case for wires and they have the same cost which leads to communication becomming the source of cost in design. A solution for cost-effective communication in todays designs is exploiting layered approach of networks for on-chip communication. In this chapter Networks on Chip (NoC) as a novel solution is presented. Different aspects of NoC as its architecture, characteristics, technology trends in NoC that may cause errors and issues related to its testing are discussed.

NoC testing (related works) 3

With integrating the whole system in a single chip, *System on Chip (SoC)*, the complexity of chip increases. On the other hand, with downscaling of technology, as discussed in previous chapter, the probability of occuring faults increases. Hence there should exist a complete strategy to test all parts of SoC. SoC testing has two separate parts; testing the existing cores on SoC and testing the interconnect structure that is responsible for connection between cores. Testing the interconnect structure of the system compromises of testing the on-chip interconnection which is either the bus system or the NoC.

Since the cores on the SoC are deeply embedded, direct accessing to their I/O pins from the I/O pins of the chip is not possible. Therefore one of the main challenges of testing the cores is to transfer the test data from test generator to the inputs of the core-under-test and also from its output to the response analyzer. This issue can be addressed either by using a dedicated TAM [1] or by reusing the on-chip interconnect which is either bus system or NoC.

For embedded core testing, a general test architecture is proposed [58,59,60]. As depicted in Figure 3.1 This architecture is composed of:

- **Source.** It generates test patterns and can be on-chip or off-chip. It is also called Test Pattern Generator (TPG).
- **Sink.** It analyzes the test responses and can be on-chip or off-chip. It is also called Test Response Analyzer (TRA).
- **TAM.** It is responsible for transferring test data from the source to the input of the core-under-test and also from the core-under-test output to the sink. It increases controllability and observability for testing embedded cores.
- **Wrapper.** It covers the core-under-test in order to connect the TAM ports to core-under-test ports. The wrapper is also responsible to switch the router's(The circuit-under-test) operational mode, between test-mode and functional-mode.

IEEE 1149 (JTAG) which is a boundary scan method is exploited for transferring test data to embedded cores and also wraps around the cores [61]. IEEE 1500 standard is developed to provide a standard wrapper that is modular and can be used with different TAMs. IEEE 1500 also standardize the test information transfer model by a Core Test Language [58].

There are quite a large number of papers about techniques used for testing cores in SoCs. In [31,32], they have proposed a special design as a dedicated TAM for using

[1] Test Access Mechanism.

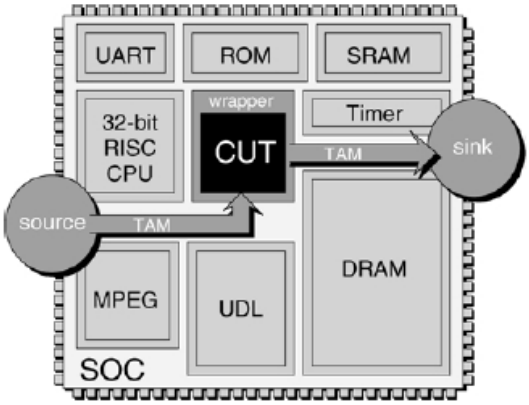


Figure 3.1: General test architecture for embedded cores [58]

in core testing. In [36], they have proposed to use bus for transferring required data for testing the core. In [33,34,35], the authors propose the use of NoC as TAM and they have designed a special wrapper to be used with NoC as TAM. Reusing NoC structure as TAM for testing embedded on-chip cores has number of crucial advantages that among them the most crucials are reducing hardware overhead related to dedicated TAM and taking advantage of high degree of parallelization in NoC to transfer test data. By reusing NoC channels for TAM, test data that targets different cores can be transfered concurrently which reduces test time. However it is crucial to note that for reusing NoC in embedded core testing, first the NoC itself should be tested.

Existing literature has enough papers related to testing bus system as an on-chip interconnection [37]. But, unfortunately due to the novelty of NoC concept itself, still there is not enough research done about its testing, which makes the application of NoC in real designs inconvenience. Therefore in this project we are focusing on NoC testing. In this chapter, previous methods of NoC testing are categorized into four categories of; offline-functional, offline-structural, online-functional and online-structural.

3.1 NoC testing

NoC infrastructure is composed of three main parts (routers, links, NIs) with special responsibilities. Since the functionalities of these components are decoupled from each other, it is possible to test each of them separately. Testing each component separately in NoC, means we are testing NoC structurally. An alternative way is to functionally test the whole NoC. In this case testing different elements of NoC is not decoupled from each other and they will be tested all together. The latter method is cheaper, but does not define where is the defected element. In what follows the previous works done for testing each part of NoC is given.

3.1.1 Testing links:

In this section the previous works done for testing the inter-switch links in NoC are discussed. Inter-switch links in NoC, as wires and interconnects at board level, more than any other fault are in jeopardy of crosstalk induced faults. As mentioned in previous chapter, as density in chip increase, distance between wires decreases and accompanied with higher functional frequency, links are more susceptible to crosstalk noises.

Other faults that can happen in links are shorts between number of links (so called bridging), opens, shorts to VDD (stuck-at-1), shorts to Ground (stuck-at-0).

For testing NoC links, we can extensively get use of testing interconnect wires at board-level, that uses the special fault models of interconnects [43]. However there are three major differences between NoC link test and link test at board level;

- Testing links in NoC is more complex than link testing at board level due to increasing complexity of SoC compared to *Printed Circuit Board (PCB)*.
- NoC links have more timing constraints and are more sensitive to delay faults, therefore they need more testing of delay faults
- Since links in NoC are not shared between different switches and each pair of switch has its own links in between, testing NoC inter-switch links are easier than testing links in PCBs.

Testing NoC links as other components can be done functional/structural or offline/online.

The original method of testing interconnects in PCB is Boundary Scan (BS)[44]. It is mainly used to transfer the test data to on-board interconnects (since the access to them is hard from I/O pins), however BS can not be used efficiently for NoC's link testing due to number of reasons listed below [43]:

- NoC has a more complex structure than links at board-level and on the other hand BS design adds additional hardware to the design. Since NoC is already complex itself, BS may not be the efficient way for NoC's links testing.
- The BS tests the system by serially shifting the test data to the design and shifting out the response data for analysis. This policy of testing is not suitable for embedded testing, since it is dependent to the architecture of each design and has to be changed or modified when the design is changed, hence it is not modular.
- Since the applying of test sequence and receiving the response should be done when the design is not in application, the BS test method is offline and can not be accomplished at-speed. But, at-speed testing is a crucial point of NoC testing, since delay faults can be detected only by the use of at-speed testing.

One of the simplest methods of testing links at board-level is exploiting *Linear Feedback Shift Register (LFSR)* for *Pseudo-Random (PR)* test pattern generator and a *Multiple Input Signature Register (MISR)* as response analyzer. Number of

test methods make use of LFSR/MISR accompanied by BS method [45,46,47,48], which makes at-speed testing of links at board-level [45,46] or at SoC level [47,48] possible. However, it should be mentioned that using LFSR for producing PR test sequence may be inefficient. This is because, they may not cover all static (short, open, stuck-at) and dynamic (delay-fault) faults. For covering all of these faults, we may need to increase the length of PR test sequence (with non-deterministic length), leading to inefficient testing. Problem with MISR as response analyzer is the aliasing problem. Aliasing means that two or more incorrect values can get XORed together and end up in a correct result [45]. In another word it means MISR may lead to mask the error and the error may be left undetected. [47] tries to solve the problems of LFSR by an algorithm which is based on graph coloring and [48] tries to solve the problem of low fault coverage of LFSR for crosstalk defects by using wighted pattern generation.

All proposed methods in [44,45,46,47,48], can be categorized as offline-structural testing of interconnect. The reason is that all of them are trying to generate or employ test vectors for error stimulation and detection in interconnect, so they are structural. They are offline because the proposed architecture for them does not let them to be tested while they are in application.

In [49], the authors are using a non-linear feedback shift register to generate the worst-case switching activity of the interconnect-under-test and then applying them by BS chain. In [50,51] authors propose designs for response analysis which can be used together with a TPG method.

The method which is proposed in [50] can be considered as functional testing. Because, it is comparing the sampled data before it enters in the interconnect and after it leaves the interconnect to see if they are the same. It can be used without a need to a specific test pattern to detect interconnect defects. This method is online because the architecture used for data sampling and comparing does not interfere with normal operation of device. While the device is being tested, it can continue to its normal job, therefore it is online.

Method that is proposed in [51] for response analysis is structural. However, they do not use test vectors to detect possible errors. Instead, they have proposed *Skew Detector (SD)* and *Noise Detector (ND)* cells that will be added to the interconnect for detecting *skew* and *noise* correspondingly. In my point of view, this method can be done online, since the added cells do not make any problem in normal functionality of the system.

In [52], the authors propose to detect the crosstalk and delay faults by measuring crosstalk noise and delay in interconnect-under-test. But this method is too expensive. It can be categorized as structural and online test.

In [53], a new structural method is applied which do not need any test vector. They detect faults by sending an impulse signal in the interconnect and then expect to receive the reflection of the impulse in a certain amount of time. If the reflection signal does not arrive until a certain amount of latency the interconnect is known as faulty. The method can be placed in offline testing methods. Since in the time

of sending pulses in the interconnect, it can not be used for transferring data.

Another series of works, propose methods for *Test Pattern Generation (TPG)* [54,55,56]. In these methods, test sequences with deterministic length are produced, opposite to LFSR which may need long test sequences. Test sequences with deterministic length should be well organized to detect all of defects related to interconnections and for this reason they should use specific fault models like MAF. *Counter-sequence*, *walking-one(zero)* and *interleaved true/complement* algorithms are examples of TPG algorithms which produce test sequences with high fault coverage related to interconnects. However, the first one is not well organized for detecting delay-faults which make it unsuitable for TPG in NoC test. methods proposed in [54,55,56] are also structural since they are proposing test vectors to be used in testing methods. They are offline, because using these methods, means that we need to send test vectors (as test packet) to interconnect to be tested. Therefore the interconnect can not be used at the test time to transfer normal data packets.

The authors in [57], propose a functional test method for testing asynchronous NoC links connecting between two switches, which are in two different clock domains. It is claimed that this method detects both static (stuck-at, short, open) and dynamic (delay faults) errors. In this method they are sampling data which is passed in interconnect before and after the *handshake signals*. If there is an error in the line, the data will not be the same before and after the handshake signals. Therefore by comparing the two sampled data they can detect the error. As we can see, They are not using any test vector and they are just examining the functionality of the link, therefore it is functional testing. This method is categorized in offline testing, since the architecture used for BIST does not let the normal operation mode and test mode to be simultaneous.

The method proposed in [43], uses deterministic length test vector which is produced by *Interleaved True/Complement Code*, therefore it is structural.

3.1.2 Testing routers

As mentioned in the previous chapter, router is composed of a combinational logic that is responsible for routing (RLB), input/output ports and buffers, as depicted in Figure 2.3. Considering this architecture, different types of faults can happen in combinational logic part and in buffer part of the router. The faults that may occur in combinational logic part are stuck-at faults, short/open, the faults related to crosstalk. The faults that are possible to occur in buffers can be due to SEU errors and cause bit corruption in the buffers.

Since errors with different roots may occur in different parts of the router, there should exist specific test pattern to detect each type of fault in the routers. This is the responsibility of the TPG to take care of generating an appropriate test pattern to stimulate the possible faults that may occur in the router. When appropriate test patterns are prepared, there should exist an appropriate architecture to transport these test patterns to the router-under-test.

Current literatures for router testing consider routers as the on-chip cores on SoC and uses the general architecture for embedded core testing in SoC (source, sink, TAM and wrapper).

The design used for TAM and wrapper deeply depends on the NoC topology. Two types of topology used in current NoC test methods are shown in Figure 3.2. In Figure 3.2 (a), the whole NoC is considered as a core that is in the middle of SoC and there is no access to its ports through the ports of the chip. In this topology only boundary routers are connected to IP cores, this is called indirect topology. In Figure 3.2 (b), the NoC is spread over the chip and has access to the I/O ports of the chip. This topology is called direct topology. In this topology, each router is connected to a core. Additionally the design for TAM always depends on the TPG and TRA, weather it is on-chip or off-chip.

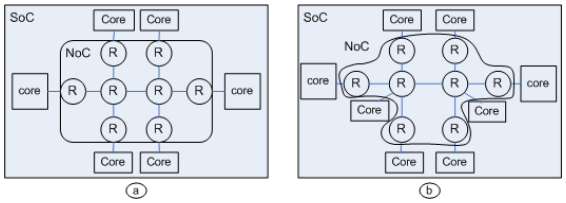


Figure 3.2: Two topologies of NoC that affect TAM and wrapper design

Each of previous works done for router testing, proposes an architecture for the TAM, for the wrapper or for both of them. There are two types of works done in this respect. One group of works are using a dedicated access mechanism for TAM and the other group is reusing the NoC for TAM. Each of these group of works are explained below.

(a) **First group: Dedicated TAM.**

- One of the first works done for router testing is [62]. They use IEEE 1149 boundary scan standard (JTAG standard) in order to transfer test data in router ports and as wrapper for a group of routers-under-test. All routers are full scan. In this way, the full scan provides test access to inside of the routers. Figure 3.3 depicts the architecture used in this method.
  In this method, first routers are grouped. Each group of routers have the same test block which consist of a *Test Access Port (TAP)* controller, boundary scan register, instruction register and other necessary logics (e.g. for decoding the instructions). All of these parts are components used in IEEE 1146 boundary scan standard. Routers in each group receive the same test data via a boundary scan register and then the responses are compared to each other by an on-chip comparator. Since each group of routers have its own test block, different groups can be tested in parallel. Therefore the test time greatly depends on the number of routers in each group. If larger number of routers are grouped the test time is less,

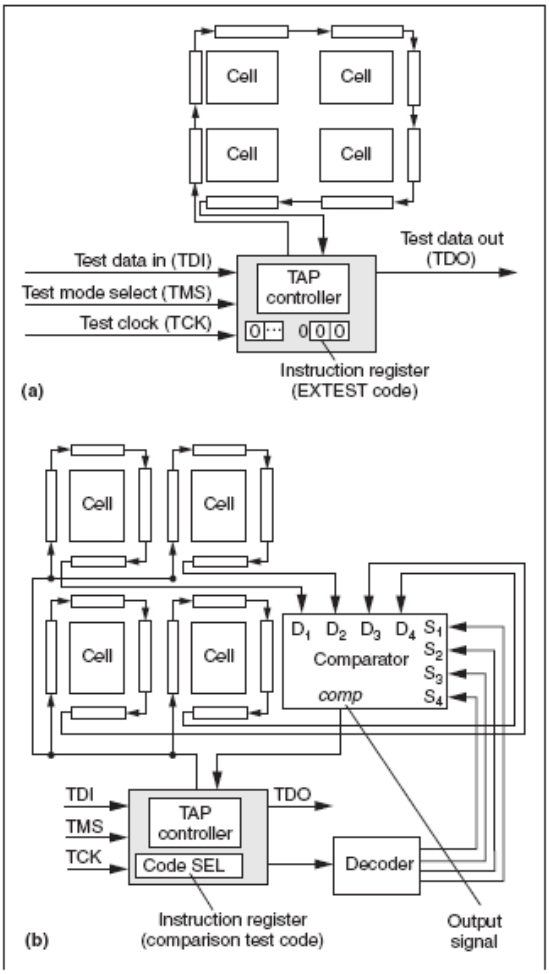


Figure 3.3: Testing architecture used in [62]

however by having more routers in each group, in case of test failure, it is harder to find which router is exactly faulty. The main drawback of this approach is long test time. This means, for a long time the NoC can not be in functional mode and can not be used in application. Moreover, they are adding extra hardware for TAM, which is a hardware overhead for system.

- In [64], the authors propose to use partial scan for each router and a single modified IEEE 1500 standard as wrapper for the whole NoC. In this method, they have used a single wrapper that covers the whole NoC , but at the same time they provide test access from network interface to inside structure (buffers and flip-flops) of each router by having partial scan chains. By partial scan chains in routers there is accessibility to the buffers of routers and to the flip-flops of combinational logic part. Test vectors (for testing buffers of routers) broadcast to all routers by a single

pin in network interface and the responses are compared by an on-chip comparator. For applying the same test pattern to functional inputs of routers (that are inputs of combinational logic part of router) , in test mode, the functional input of router receive test patterns from Ci_x , as depicted in Figure 3.5.

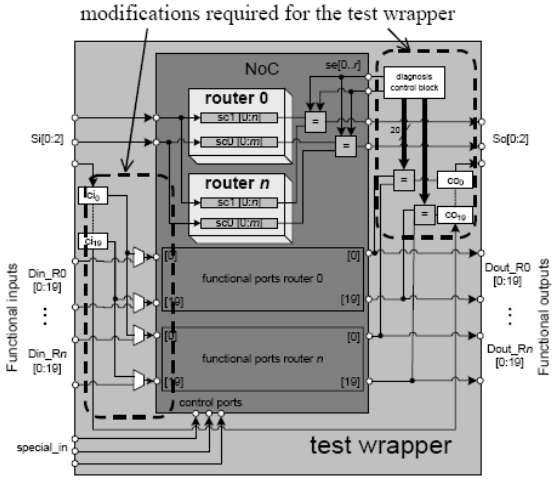


Figure 3.4: Wrapper design used in [64]

In [64], In case of test failure each router can be tested independently and test data will be distributed among them. The modification they did in the wrapper IEEE 1500 is that instead of having equal number of C_is to functional inputs of NoC (which is the sum of functional inputs of routers) they apply C_i cells only equal to the number of channel bit-width. They broadcast the same data for each channel bit to all functional inputs of all routers. The modification they did in the wrapper leads to reduction of area overhead and also test time (by reduction in the number of shift operations.). The drawback of the proposed method in [64] is that for large scale NoCs that have large number of routers, the fanout of Ci_x input pins and scan input pins become large. Moreover there is no method given as TAM for transferring data from source (test pattern generator) to input of network interface and the test is offline.

(b) **Second group: NoC reused as TAM.**

- In [63], they propose to reuse NoC as TAM for transferring test data to its own routers. For this reason they assume that the links between routers and network interfaces are tested before and they are fault free. In this method, they group routers based on their accessibility via the input ports of the chip in an ascending order in a way that the routers that are closer to the input ports of the chip has the smaller order number. Figure 3.4 represents this configuration for router testing.
  They start testing routers with the smallest order number in each step. The routers in the same group can be tested in parallel. In this way

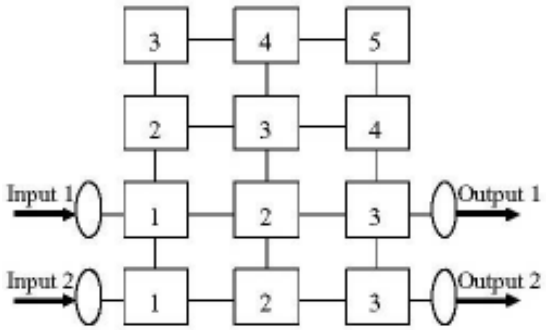


Figure 3.5: testing configuration in proposed method in [63]

they are assured about the TAM they are using for transferring test data to router-under-test. The wrapper used in this method is IEEE 1500 that covers each router and the connected core to it. The wrapper can configure each router or each core connected to it, to test mode or functional mode. If the router is in the way of transferring a test vector to a router-under-test (or core-under-test), it will be in functional mode and if it has to be tested, it is in test mode. The method they used for test response analysis is on-chip comparator which compares the responses of the routers that are in the same group of order number. However they do not provide information about the comparator that when a test failure occurs it can be used for diagnosis of the exact router that is faulty. Using this method for TAM leads to reduction of hardware overhead, test time (due to using high bandwidth channel and parallel communication of NoC for transferring test data) and number of additional pins in comparison to dedicated TAM like boundary scan, as it was used in [62]. However, in this work they do not consider buffer testing of routers that need additional scan chains to be tested and they only talk about combinational logic part of the routers. Moreover, the test in not done online. When the routers are being tested they are not used anymore for transferring functional packets.

- In [59], method for testing an asynchronous NoC is proposed. They reuse NoC as TAM and they employ a modified version of IEEE 1500 as wrapper for routers. In asynchronous NoC, routers might be in different clock domains. For this reason they propose that each router has its own wrapper which are asynchronous to each other. Hence, they have to modify the IEEE 1500 for having handshake signals. In this architecture they introduce another unit, named as Generator-Analyzer-Controller (GAC), which plays the role of source (TPG) and sink (TRA). It can be off-chip or on-chip. Additionally, it has the role of controlling and configuration of asynchronous wrappers. Extra components that they have used in their method are additional channels that transfer the data needed to configure

the wrappers by GAC. It should be noticed that, in addition to testing inter-router links, first it is needed to test the configuration channels as well. This is an overhead in test time. This method is similar to the method proposed in [63] that is for synchronous NoCs.

- In [65], they propose to reuse NoC as TAM for full scan chained routers. They have proposed a special design for the wrapper that covers each router. The main idea of the method is reusing NoC to transport test data to routers. For this reason, if we consider the network of routers as a graph, the first node that should receive the test vectors from source should be defined. For this reason by employing *Dijxtra algorithm* they find the router in the network that has the shortest path to all other of the routers, the so called topological center. When test vector is inserted in the topological center router, it sends the test vector to its own scan chain and also broadcasts it to the next available group of routers. In addition it sends its test response to the next group of routers, so they can compare it with their own responses. All routers repeat this procedure except boundary routers. For realizing this method special wrapper is designed that is depicted in Figure 3.6.

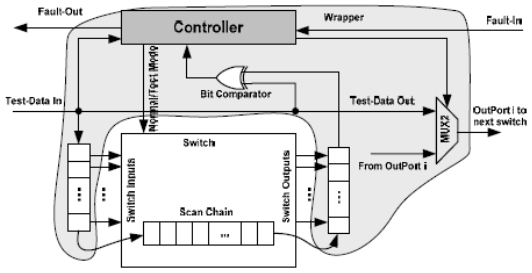


Figure 3.6: Wrapper design used in [65]

This wrapper has scan chains for input and output ports of the router, controller and comparator. This is comparable to the method proposed in [63] that they also reused NoC for transferring test data to the routers. However In [63] the nearest routers to inputs of the chip (that are from boundary routers) are the first receivers of the test data. The advantages of method [65] over [63] is that it is independent of the NoC topology and the test time is less since the test data is inserted to the topological center of the NoC. The disadvantages compared to [63] is larger hardware overhead in the wrapper in comparison to IEEE 1500 which is mainly due to scan register in the input and the output ports of the routers. In this method also, when routers are being tested the NoC can not be used to transfer normal functional packets. Therefore the test is offline.

- In [66], they target both combinational logic part of router and testing of buffers in routers. For combinational logic test, they use off-chip source from an ATE machine and they propose reusing of NoC as TAM as in

[65] and [63]. They use one boundary router to insert test data to NoC. They propose to compare the results of the routers, so there is no need for off-chip or on-chip response analysis. For testing buffer part, they use full scan and they propose a special march test based on possible faults that may occur in the buffers. The BIST architecture that they use for buffer testing is composed of a shared source and controller for the buffers in all routers and a local response analyzer (sink) for each router that at the end they are derived by a MISR to produce a signature to be analyzed. This scheme is showed in Figure 3.7. The reason they used the distributed response analyzer is that it has a smaller area overhead than the centralized response analyzer.

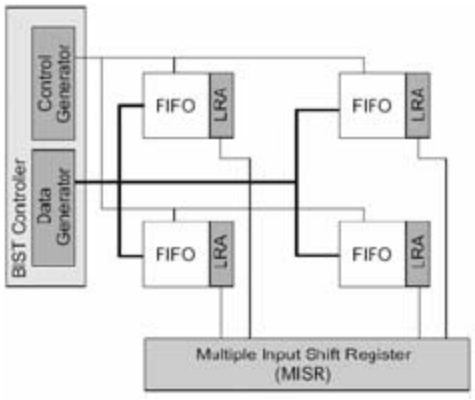


Figure 3.7: BIST architecture used for buffer testing in [66]

Functionally testing routers means that we should check if the router connect each of its inputs to its correct output port as it is expected in the address field of the packet that is being routed through the router. Otherwise if we check the possible errors that may happen in each part of the router by sending test vectors to stimulate these faults, we are doing structural testing.

All of the methods discussed above for router testing are structural, since test vectors are being sent to the special part of the router to stimulate the possible faults in those parts and check if they behave faulty or not. In terms of online or offline, methods discussed above are doing router testing offline since while the routers are being tested they can not be used for their normal operation which is routing data packets in NoC. Therefore testing the NoC would be offline.

In [59,62,63], they use Figure 3.2 (b) topology and in [64] the authors have used Figure 3.2 (a) topology. If the method proposed in [64] (which uses a single wrapper for the whole NoC) has to be used in the topology of Figure 3.2 (b) (that each router is connected also to an IP core), the wrapper has to support also cores.

3.1.3 Testing NI

In the current literature, there is no testing method specific to NIs. In [10] they propose to test the NIs with the attached core to them. This method is functionally testing the NIs, since we test is the NI can provide the expected functionality for the attached core or not. This method does not provide a high fault coverage for the NI and does not give detail information of the defected parts of the NI. In [42], DFT techniques are proposed for testing NIs. Testing NI through DFT, can be a structural test with high fault coverage. However, DFT techniques can have high area overhead. Moreover, since NIs usually have a complicated architecture, test pattern generation is hard.

3.2 Fault tolerant techniques used in NoC

In order to achieve a fault tolerant NoC, 4 general techniques can be used:

(a) **Error detection and re-transmission.** In this method, by using error detecting codes, added to data packets, faulty packets can be recognized. If a packet is detected as faulty packet, it will be retransmitted. One of the previous works that uses this method is [40]. In [40], they have added error detection codes to the data packets. For detecting a fault, they use encoders and decoders. Checking for a possible fault can be done either switch-to-switch or end-to-end. Whenever a faulty packet is detected, it should be retransmitted. However the problem with this method is when there is a permanent fault or when there is a SEU in the routers. When there is a SEU in a router, it means one of its internal flip-flops is corrupted. This results in the router entering a faulty state. For recovering from this situation only restarting the router from the initial state can help. In other words in this way retransmission of the packet does not help. Because the router has entered a faulty state and on the next clock cycles will enter to another faulty state and again does not route the packet to the correct destination.

Moreover, when there is a permanent fault in one of the routers in the path of the packet, the problem can not be solved by retransmission. Even, it can retransmission of the packet, again and again may lead to deadlock.

(b) **Error correction.** Error correction like the one used in [41] is more effective. Since they do not repeat the packet transmission from source to destination. They just add error correcting codes to the packet. In this way there is no need to detect the fault and do recovery after finding it. But, it should be noticed that this method uses information redundancy that is used in error correcting codes. Information redundancy also involves hardware redundancy which is considered as an overhead of this fault-tolerant technique. This hardware redundancy is due to number of bits added in order to store check part.

(c) **Fault masking.** Fault masking techniques like Error correction is effective and does not have problems of error detection and re-transmission. In [38],

they mask the fault by methods like triplication and voting.

(d) **Error location and re-routing.** By error detection and locating (e.g. define which router or link is faulty), the error can be recovered by rerouting and excluding the faulty element. Since the error detecting and locating is used to make the system fault tolerant, it should be accomplished while the system is performing its normal functionality. In other words, we should use online testing to detect the fault. Therefore, while we are testing a module in NoC, it should continue to its normal functionality that is packet transmission between on-chip cores.

In [39] they propose a method for online detection of faulty links and routers, by using a self-checking circuitry and coded inputs/outputs. They do parity checking at the input of the routers, as well in the outputs. If an error is detected in the input, it means the link connecting to the input port of the router is faulty. But, if an error is detected in at the output of the router, it shows that the router is defected. In this method, they are functionally testing the router. In other words, they just check if the router transfer the expected packet to the expected output. They are not sending special test patterns to the router in order to test the structure of the router. Therefore after testing the router, we just know if the router is faulty or not. We can not achieve to the detail information on which part of the router is defected.

3.3 Summary

In this chapter, previous works on NoC testing is discussed. This work is devided in two groups of testing NoC links and testing NoC routers. For better understanding strong points and weaknesses of each method, they are categorized in 4 groups; functional-offline, functional-online, structural-offline, strcutural-offline. Works done for NoC link testing were either structural-offline or functional-online. Works done for router testing were all structural-offline. The future goal of testing NoC is structural-online testing. In next chapter we will focus only on online testing.

Platform for NoC Online 4

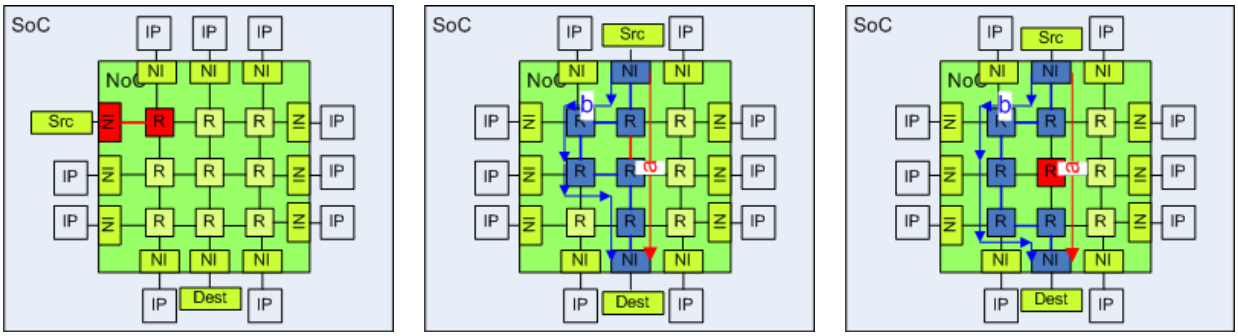
Reduction of VLSI feature sizes and increase in transistor density in modern VLSI integrated circuits, on one hand increases fault rate and on the other hand increases data path resources and on-chip connection. Increase in data path resources and on-chip connection makes the online testing and fault tolerance features more cost-effective.

By using online testing low latency fault detection, localization and recovery should become possible. As mentioned in previous chapters, in a NoC based system, NoC has the critical responsibility of communicating cores and at the same time, because of its specific structure, it is in danger of more faults. Also in NoC there is inherent hardware redundancy. Therefore, by knowing these characteristic of NoC and the above mentioned points about online testing, online testing in NoC seems to be worthy and promising.

Due to what was reviewed in previous chapter,error detection and rerouting is one of the effective methods to make the NoC fault tolerant. This method works well for NoC since NoC is composed of a number of identical elements. If one of the elements is faulty it can be replaced by one or more identical components. In other words, the path from a source to a destination will be exchanged with a new path that still connects the source to the destination.

For this technique there is a need to diagnose the fault. In other words we should be able to determine which router, NI or link is faulty. In case that a non-boundary router (a router that is connected to NI in-directly via another router) or an inter-router link is the faulty element the error can be recovered by rerouting and changing the path from the same source to the same destination. However, if one of the boundary routers (the routers that are connected to the NIs directly) or the NI or a link between NI and router is faulty, the problem can not be solved online by changing the path. This is because each core is attached to one and only one NI. That makes the NIs the only possible gateway to the IP cores. This issue is depicted in Figure 4.1. Figure 4.1.a shows that if a NI, a link between NI and router or a boundary router is defected (shown in red color), there is no other element that can be replaced. In Figures 4.1.b and 4.1.c an inter-router link or a non-boundary router itself are the defected elements (defected elements are shown in red). The error can be recovered by rerouting which is shown by the blue path (path b) from source to destination instead of red path (path a).

Therefore error detection and rerouting is an effective method only for non-boundary routers and links. Because, in other cases (NI, boundary router or link) the problem can not be solved by rerouting.



(a) Defected NI, boundary router or boundary link

(b) Defected inter-router link

(c) Defected router

Figure 4.1: rerouting possibility in different cases of faulty element

The purpose of testing the routers or links in this method is to make the system more tolerant against possible faults. This means that while the system is functioning, if an error occurs the system can still continue to its normal functionality. Therefore, inn this method, the routers or the links should be tested online without halting the normal functionality of the system. In this project the focus is on online testing of the routers.

In previous works testing the routers are either online functionally or offline structurally. The result of online functional test informs us about correct/erroneous functioning of a router. This information is enough to use the method of error localization and rerouting, Because it shows which is the faulty router in case of an error. However, if we want to know the reasons that the router is not functioning correct, we have to test the router structurally. In this way the defected flip-flop or gates of the router can be defined which can help us later in fault diagnosis.

In order to do structural test on a block, as mentioned in first chapter, the block can not continue to work in its normal mode. Rather, it should change to test mode. In other words, when a block is being structurally tested , it can not be done online.

But online testing of the NoC is a different case, due to its inherent block redundancy (prosperity of identical elements). It is possible to structurally test the elements of the NoC, while it is accomplishing its normal functionality which is transmitting packets between on-chip cores. In next section a method is proposed to do online structural test on NoC.

There are Two different approaches for testing NoC. The whole NoC structure can be considered as one of the cores of SoC like other cores, or alternatively, we can test each part of the NoC separately, e.g. consider each router of NoC as an individual core. Therefore for testing NoC the same general architecture as Figure 3.1 for testing embedded cores can be applied. For example we can either wrap the whole NoC with a wrapper and define a TAM to transport test data to the NoC or we can wrap each router with wrappers and define a TAM to have access to the routers. The two alternative approaches are shown in Figure 4.2.

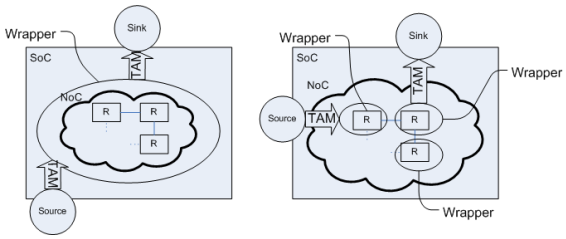


Figure 4.2: Two general architecture for wrapper and TAM in NoC

Considering the whole NoC as a single on-chip core, can not be useful when we want test the NoC online. Because NoC as a single core can be either in test-mode or functional-mode. This means the test is offline. Therefore for online testing we consider each of the routers of the NoC as a single core. In other words, each router will have the same test architecture that has been proposed for testing on-chip cores. In this project, the routers are considered as on-chip cores. Therefore the same test structure will be used to test them; TPG, TRA, TAM and wrapper. As in embedded core testing the test pattern should be provided by core developers, in this project is it assumed that the test patterns for routers are provided by the router developers.

4.1 Overview and flow

By taking advantage of router redundancy in NoC, we can do structural test on routers without halting the on-chip communication. In other words, NoC will be tested online and structural. When a router is being tested, the NoC exclude it from the list of available routers and set it to test mode. If the excluded router is part of the path that a packet should go through, the packet will be rerouted. It means the router-under-test will be exchanged with one or more routers. The general architecture of a NoC and the mentioned method is depicted in Figure 4.3. It is obvious that, in this way, the NoC can transfer the data packets between on-chip cores, beside testing a specific number of routers.

4.1.1 What is needed to implement the proposed method

In order to realize this method for online testing of routers, there is need to a proper test architecture for the routers and a high level test scheduler.

4.1.1.1 test architecture for router testing

In order to implement this method, first of all the test architecture for testing each router should be defined. In this method, each router of the NoC is considered as

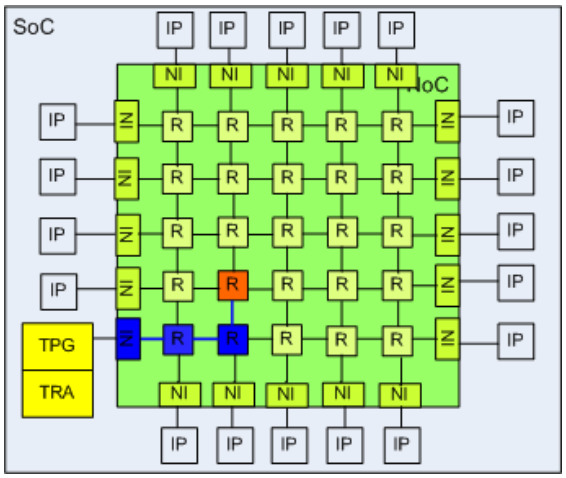


Figure 4.3: General test architecture for NoC routers

an individual on-chip core. Therefore, the conventional test architecture for testing on-chip cores will be used for routers. The test architecture is composed of:

- **TPG (Test-Pattern-Generator) and TRA (Test-Response-Analyzer).** TPG will generate a proper test pattern, depending on the router architecture. The test patterns should be able to stimulate possible errors in combinational part and buffers of the routers. TRA is responsible for analyzing the test responses. Since the test is online, the TPG and TRA should be on-chip. Therefore, TPG and TRA can be considered as one of the on-chip cores. TPG and TRA design is not addressed in this work.
- **TAM (Test Access Mechanism).** There are 2 possibilities for the mechanism that the test data will be transfered to the router-under-test. the conventional method is to use a dedicated channel as TAM. But it has hardware overhead and the dedicated channel may not have the capability of transferring test data in parallel which increases the time that test data is transferred to the circuit-under-test. Another alternative is to reuse NoC as TAM to transfer test data to its own router. This arrangement has the advantage of eliminating extra dedicated channels for transferring test data. Moreover, NoC communication fabric has characteristics that can be helpful for test data transmission, such as QoS and high degree of parallel transmission.

  In this project the second method, which is reusing NoC as TAM, is employed. The detail information about how NoC is able to transfer test data to its own router will be discussed in section 4.3.
- **Wrapper.** The wrapper should cover the router. The responsibilities of the wrapper are connecting the router which is in test-mode to the NoC, transforming the test packets to streaming test data and vice versa and changing the router to test-mode or functional-mode. The wrapper should be transparent in functional mode. This wrapper has to be designed.

4.1.1.2 High-level scheduler.

In addition to test architecture for testing the routers, to implement this method, there should exist a scheduler that defines at each time, which router is in test-mode and which router is in functional mode.

Since we wanted the method to be modular, the design should be somehow that there is no need to change the design of the routers or NIs in NoC. Therefore test packets should have the same format as data packets. Otherwise we need to change the design of the NI. Because, these are the NIs that packetize and de-packetize the test data and test packets. If the test packets had to have different format than data packets, we had to design special NI (that is connected to TPG-TRA) to packetize and de-packetize the test data. Hence, the test packets does not have any special bit to show that they are different than data packets and they should be treated different. When a packet arrives to a router, the router does not recognize if it is a normal data packet or a test packet. This is the reason that there is a need to know at each time which router should be in test-mode and which router should be in functional-mode. The high-level scheduler is providing us with this information.

The inputs to the scheduler are:

- The available routers: available routers are the routers that are not in test-mode or defected. Therefore, they can be calculated by subtracting the list of defected-routers and routers-under-test from the list of all routers in NoC.
- The required connections between IP cores on the chip that a running application on the SoC needs. These connections should be known in order to define the busy routers in NoC that should be in functional mode.
- The already tested routers. In order to send test packets to the routers under test, we should use the routers that are already tested. This is due to the fact that NoC is being reused as TAM. This issue and the order of testing the routers will be discussed in more detail in section 4.3.
- **Ready vector.** A vector that has the size equal to the number of routers, is the input of the scheduler. Each bit of this vector defines whether the corresponding router to it, can go out of test mode or not. The signal ready corresponding to each router is produced by its wrapper. This signal is needed in order to be able to close the connections between the corresponding wrapper and the attached NI to TPG-TRA. This issue will be discussed in more detail in next chapter.

An important point is that the connections that are needed to transfer the test packets to the routers-under-test should also be added to the list of required connections on NoC. In order to provide the required connections for transferring test packets, the scheduler uses only the routers that are in the already-tested-routers list.

The output of the scheduler is the id of the routers that should be tested at each time. The id of the next router-under-test will be as input to the connected NI

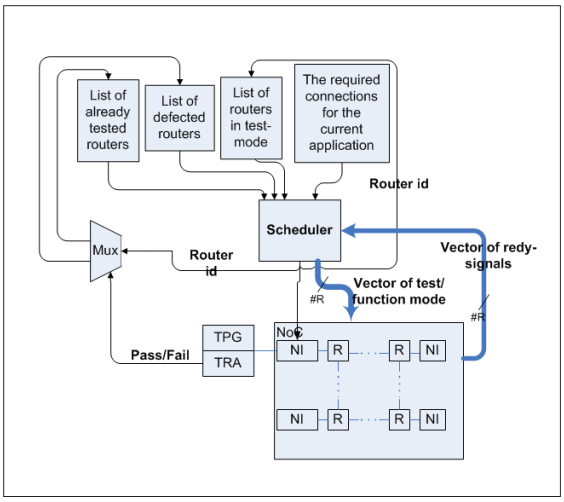


Figure 4.4: High-level scheduler in the system

to the TPG-TRA. Moreover, a vector with equal size to the number of routers is generated as output. Each bit of this vector goes to a router and makes the router in test/functional mode. Figure 4.4 depicts, how the scheduler should be connected to the rest of the system.

4.1.2 Phases of testing routers

The router will be tested in 2 main phases, that each phase is composed of number of other steps:

(a) testing the sequential part of the router (The internal flip-flops and buffers).
   i. Test vector should be sent from TPG to the scan chain of the router.
   ii. Test response should be received from the scan chain of the router to TRA.

(b) testing combinational part of the the router (the routing and arbitration logic).
   i. Router state should be initialized by sending test vector from TPG to the scan chain of the router.
   ii. Test pattern should be sent from TPG to the functional input of the router.
   iii. Test response should be received from functional output of the router to TRA.
   iv. Router state should be received from the scan chain of the router to TRA.

testing the router in tow phases of sequential and combinational parts, helps us with fault diagnosis, in case we want to fix the problem later. If a proper test patterns are applied, it is also possible to define which flip-flop or which gate is

the defected one. For testing the sequential part, a special test pattern will be generated to test the flip-flops. It is supposed that the internal flip-flops of the router are scan flip-flops and are chained together. The test vector is inserted to the router via the scan-in input and will be received from the router via the scan-out pin. After assuring the correct functioning of the internal flip-flops, it is time to test the combinational logic of the router.

For testing the combinational logic part of the router we need two types of test data:

- Test vector for initializing the internal flip-flops.
- Test pattern to send as stimuli to functional input of the router.

First the internal flip-flops should get an initial value, in order to give an initial state to the router in test mode. Therefore, the test vector for initializing the internal flip-flops should be inserted to the router by scan-in pin.

After router sets to the initial state, the test stimuli should be applied to one of the functional input ports of the router. Depending on the functionality of the router when it is in test mode, after some time the test response will be ready. The response will be received from one of the functional output ports of the router and sent back to the TRA to be analyzed.

After the test stimuli has applied to the router, the content of the internal flip-flops define the current state of the router. The current state of the router also has to be analyzed to see if the router has entered to a legal state after test stimuli has been applied or not. Therefore, they should also be scanned-out and sent to the TRA.

TRA will compare the received test response and state of the router after test with the expected test response and state. The router will be tagged as defected if there is a difference between the expected values. The TRA has an output signal (Pass/Fail), as depicted in Figure 4.4, that selects if the router that was under test should be added to defected-routers list or to the already-tested-routers list.

4.1.3 Limitations of the proposed method

The key feature of the proposed method for testing the routers online, is to use rerouting when a router is in test-mode and can not route packets. However, it should be noticed that this is not the case for the boundary routers (The routers that are directly connected to the NIs.) As it is depicted in Figure 4.3, in this work, it is supposed that the NoC has the direct topology. If a boundary router is under test and is not in its functional mode, there is no chance to replace it by one or more neighbor routers. This is also depicted in Figure 4.1.a. In other words, if one of the boundary routers is under test, the connected IP core to the corresponding NI can not receive or transfer a packet. Therefore this method of online testing is more advantageous for large NoCs. In other words as much as the ratio ($r = non - boundary routers / boundary routers$) increases, this method of testing becomes more close to online testing.

4.2 Communication protocols AEthereal

Depending on NoC structure, in different parts of the NoC, data that should be transfered has different formats. In AEthereal, NI consists of 2 parts; NI shell and NI kernel. Data that is moving through the NoC between routers should have packets format. Data that is being transfered between NI shell and NI kernel has the format of a simple message. It consist of the message, a valid signal and an accept signal. And finally data that want to enter/exit NoC through NI shells, should have standard IP protocol formats (such as DTL, AXI and etc).

4.3 Reusing NoC as TAM

By reusing NoC as TAM, it is possible to employ some of the NoC features to transport test data. The most useful features of NoC are parallel transactions and QoS. Since NoC can have support number of transactions at the same time, the time that test data is transfered to the router-under-test decreases. Using QoS is test data transmission will be discussed in more detail in this section. On the other hand, reusing NoC as TAM (specially for testing its own routers) brings a set of limitations that we should take them in to account. The necessity to packetizing test data and ordering the routers-under-test are the limitations.

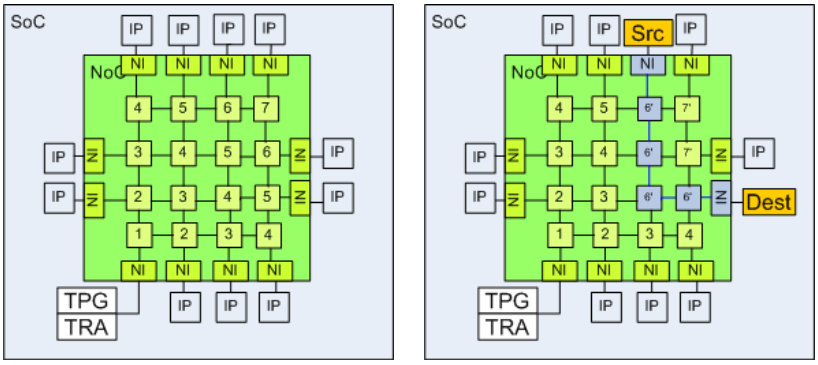
4.3.1 Order of testing routers

Since the NoC is reused as TAM to test its own routers, it is crucial to be assured about correct functioning of the mechanism that the test data is being transfered with it. In other words, in order to reuse NoC as TAM, the NoC should be fault-free. This means that the NIs, links and routers that are used to make a path from TPG to router-under-test and from router-under-test to the TRA should be fault-free. In this project we assume that the links and NIs are tested before and they are fault free. But the goal is to test routers themselves. Therefore the routers that may be in this path should have been already tested. The solution is to test the routers in order.

Routers will be assigned to an order number. The order number of each router is based on the distance of the router from the NI that is connected to TPG-TRA. The closest router to the connected NI to the TPG-TRA has the lowest order number and as the routers get farther their order number increases. Then, the routers with lower order number will be tested first. In this way the router that is attached to the connected NI to TPG-TRA will be tested first. At each time, we are sure that the routers that are included in the path from TPG-TRA to router-under-test have been tested before and are fault-free. Figure 4.5.a depicts an NoC with ordered routers.

Ordering the routers, based on their position in the NoC, is the responsibility of the high-level scheduler. However, the scheduler should also consider the current

application that is being run on the NoC and the connections between IP cores. This means that if an application is running on NoC and some of the routers are busy in providing connections for the application, their order number will increase. Increasing order number means that the router will be tested later. The ordered routers without any application on the NoC and the ordered routers with an application with one connection between two IP cores are shown in Figure 4.5.b.



(a) ordered routers without any connection in NoC

(b) ordered routers with a connection in NoC

Figure 4.5: Ordering of routers

In figure 4.5.b the order numbers defined by *prime* are the new order numbers of the corresponding routers that due to the connection between source and destination their order number has been changed.

4.3.2 Data format conversion

Since test data is being transfered via NoC, it should be packetized in order to be able to move through the NoC between routers. As depicted in Figure 4.3, TPG-TRA blocks are connected to NoC via a NI.

Test data is generated by TPG in the form of streaming serial test data. The serial test data then has to be converted to DTL format data. But in order to keep design simple, it is also possible that test data enters NI kernel directly. In other word, it is only needed to parallelize the serial test data to parallel pieces with sizes equal to word sizes. The streaming test data then has to be changed to packet format. Transforming streaming test data to packet in done by the NI.

NI will packetize the test vectors like normal data that is transmitted between cores. In addition to test data, The NI should receive the id of the current router-under-test. The NI receives the id of the current router-under-test from the high-level scheduler. The NI, by looking at the current traffic in the NoC and available routers decides for the shortest path to the router-under-test. This path will be added in the header of the test packet.

After packetization, test packets will be passed to the first router connected to the NI (which is connected to the TPG) and depending on the path that is in the header of the packet they will be routed to the intended router-under-test. When a test packet is received at the router-under-test, it should be de-packetized to streaming test data and then be applied to the router-under-test. This issue will be further discussed in section 4.4.

In return, test response should also get packetized to be transferred to the TRA. This means that the wrappers around the router should support parts of the functionality of the NI to packetize/depacketize the test data.

The return path to the TRA should also be calculated. But it should be noticed that the router-under-test may be in the middle of the chip. Therefore the wrapper around the router-under-test is not accessible by the scheduler and the return path can not be received from the scheduler. Therefore the return path should be calculated inside the wrapper itself. Since the TPG and TRA are both connected to the same NI, the return path is the reverse of the initial test packet path. The return path will be added to the header of the response test packet.

When test response packets reaches to the connected NI to the TRA, they get de-packetized by the NI. If the router is defected, the NoC will add it to the list of defected routers and retain it out of the list of available routers.

4.3.3 Using QoS of the NoC

Different levels of QoS may be supported by the NoC. The most commonly used levels of QoS in NoCs are *Best-Effort* (BE) and *Guaranteed-Service* (GS). If a NoC supports packet transmission in BE, It is sure that the packet will reach to the destination. But performance issues such as latency and band-with can not be guaranteed. If NoC gives GS service for transmission of a packet, it means that it is assured that the packet transmission meets the performance criteria such as bandwidth and latency.

Test patterns are generated as serial streaming data by TPG. When test response is being analyzed, it should also be consumed by TRA as streaming test data. Similarly, at the circuit-under-test test vectors should be applied like streaming data and also received as streaming data. Streaming data means that the test data should be continuous and not be disconnected.

Thus the challenge of the mechanism that is used as TAM, is to be able to transfer test data to circuit-under-test and TRA as streaming data. When NoC is reused as TAM, if a test vector is larger than the size of packets in a NoC, it should be packetized in N number of packets. This N packets should be transferred to the router-under-test one after another without an interruption in between. So they can get de-packetized in the destination and be applied to the router-under-test in streaming data fashion.

When NoC gives GS service to a connection it can transfer data in that connection as streaming data. In GS connection it can be assured that test packets that contain

different parts of a test vector will reach to router-under-test one after another without an unexpected delay. However this is not the case for the connections that receive BE service. In BE service, there is no guarantee for the time that a packet of a connections will reach to the destination. Therefore, in case there is a test packet containing test vectors, within a BE connection it is not defined the arrival time of the test packet to the router. when test packets are transfered within BE connections, there may be time intervals between test packets that contain different segments of a single test vector. Therefore in case BE connections for test packets, there should exist a mechanism to freeze the router-under-test until the next test packet that contains next part of the test vector arrives. Otherwise the content of the internal flip-flops that are chained together will be lost and test will fail. Two possible solutions for halting the router-under-test are:

- **Hold-able flip-flops.**Making the internal flip-flops of the router hold-able. In hold-able flip-flops when the signal hold is active the flip-flops keep their previous content.
- **Clock gating.** It is also possible to *AND* the clock signal of the router with a hold signal. Hold signal is active low and when it is low the input clock to the router will become low.

Hold signal should become active whenever there is a interrupt between receiving two test packets that belong to a single test vector.

In previous works that are reusing NoC as TAM mostly GS connections are used for transferring test packets. This is quite logical since with GS connections there is no need for hold-able flip-flops that are more expensive or clock-gating method that may interfere with the clock tree design of the system. But, with considering the disadvantages of other methods than using GS connections for test packets, again in this project it is decided to use BE connections to transfer the test packets with one the above mentioned techniques for making the test data continuous. This is due to the fact that, In this project the goal is online testing of the NoC. Therefore it is not desirable that testing the routers interfere with the application that should be implemented by NoC. It is intended that testing the routers has the least influence on the normal functioning of the NoC. Therefore, in a NoC that support both BE and GS connections, it is preferred to use BE connections for testing purpose and leave GS connections for the functionality of the NoC for serving the connections between IP cores.

4.4 Specified requirements

As it was discussed in section 4.1.1, the requirements for the proposed online test method is a proper test architecture and a high-level scheduler. In the test architecture , we assume TPG and TRA are given and NoC is being reused as TAM. Therefore it is only needed to design the wrapper. Furthermore, a high-level scheduler should be designed. Specific characteristics that the wrapper and the high-level

scheduler should have is given below. However in this project only the design of the wrapper will be addressed.

4.4.1 Wrapper design

In this section we classify the problem of designing the wrapper. The inputs that for designing the wrapper should be known are:

- The architecture of the router and the packet format of a NoC.

The output of the design is a wrapper with the capabilities listed below:

- The wrapper should connect the router while it is in test mode to the NoC.
- Wrapper should packetize/depacketize test data to/from test packets.
- Wrapper should convert the simple test messages to serial test vectors when internal flip-flops are being tested.
- Wrapper should control at each time the test data should be directed to which port of the router (functional port or scan pins).

The constraints that the wrapper should be satisfied are:

- The wrapper should be modular. This means it should not change the architecture of the NoC (including it elements, such as NIs and routers).
- It should keep the test data streaming without an interruption.

The assumptions that are made in designing the wrapper are:

- The high-level scheduler is available.
- At each time only one functional port of the router is connected to the wrapper.
- The functional port of the router that should be connected to the wrapper is defined. Because this may vary from one router to another, depending on the position of the router in NoC.
- At each time the wrapper knows if it should be in test-mode/functional-mode. In functional-mode the wrapper around the router is transparent. This is defined by a single signal test/function that is produced by the high-level scheduler.

4.4.2 High-level scheduler

The inputs that for designing the wrapper should be known are:

- The topology of the NoC.
- The NI that is connected to the TPG-TRA

- The connections needed by an application that is being run on the SoC.

The output of the design is a scheduler that receives 4 lists as input as depicted in Figure 4.4 and produces 2 outputs. Based on the received information from the input lists the scheduler should put the largest number of routers in test-mode.

Implementation and evaluation

5

In this chapter the steps to design a wrapper that makes it possible to do online testing on the routers while NoC is reused as TAM, will be addressed. As discussed in previous chapter while defining problem statement, some factors of the NoC that is being tested should be known. In what follows, first the given factors for designing the wrapper in this project will be explained. The rest of the chapter will explain the top level design of the wrapper and also the design of important blocks in the wrapper.

5.1 Router interface, message and packet format in AEthereal

The factors that should be known for designing the wrapper are the router interface architecture, the packet and the message format in the NoC. The target NoC in this work, that its routers will be tested is AEthereal NoC. Therefore as given factors in designing this wrapper we have the AEthereal router interface and the message and packet formats in this NoC.

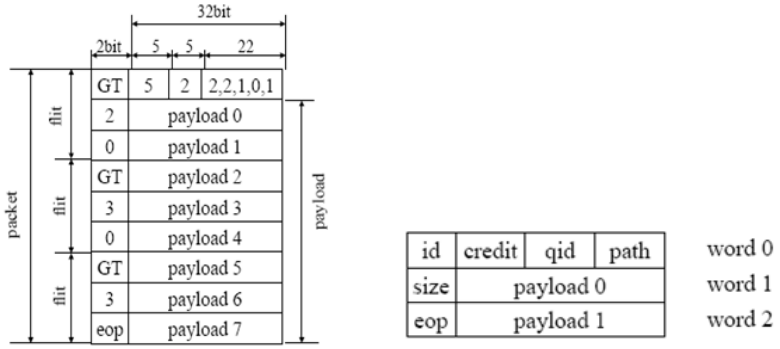
5.1.1 Packet format

The packet format in AEthereal NoC is depicted in Figure 5.1.a Each packet is composed of smaller units called flits. Each flit itself is composed of number of *phits*. Each *phit* is the actual physical data units that construct the packets. When a packet is being transferred, it will be transferred *phit* by *phit*. Each flit is composed of three *phits*.

The first flit of the packet is the header flit. The header of the packet should have different fields of information. Since the routing in the AEthereal is source-routing the path from source to destination is calculated and added to the header of the packets. Therefore one of the fields in the header is the complete path from source to destination.

In Aethereal implementations there are two different ways that the path from source to destination is formed:

- In the header of the packet, it is exactly defined that what number of bits is related to selection of the queue in the NI. In other words there is a fix number of bits that is dedicated to queue selection in the receiving NI. For example, in the current implementation of the AEthereal the first 22 bits of the first



(a) Packet format (b) Flit format

Figure 5.1: Packet and flit format in AEthereal

phit in the header flit of the packet are giving the path through the router network and the next 5 bits ([26:22]) defines the $Queue_{id}$ of the receiving NI that the packets should reach.

- In the header of the packet the number of bits that defines the $Queue_{id}$ of the input queue in receiving NI is not defined. In other words the number of bits for $Queue_{id}$ selection is dynamic and it depends on the number of ports in the NI. Therefore, there is not a fix number of bits in the beginning dedicated to $Queue_{id}$ selection. Instead, depending on how many ports the NI has, the number of bits for $Queue_{id}$ selection will be defined. In this case the number of bits for $Queue_{id}$ selection will be $\log(Number - Of - Ports)$. As an example, if the NI has 3 ports, after the first 22 bits in the first phit of the header, the 2 next bits ([23:22]) define the $Queue_{id}$ of the receiving NI.

In this project the first implementation is considered.

Moreover in AEtheral there are two types of buffer management schemes:

- link-level flow control : is between consequent hops in the path. Flow-control signals between routers themselves and between router and NI take care of link-level flow control.
- end-to-end flow control : is between the sender NI and the receiver NI. Space and credit counters in the sender NI and in the receiver NI will take care of this. The sender NI has the amount of available space in the receiver NI in its space counter. whenever it sends a packet to the receiver NI, it decreases the value of space counter based on the amount of data that has been sent to the receiver NI. In receiver NI there is counter called credit. The credit counter in the receiving NI has the amount of freed space in the receiving NI. For example if a data has been loaded out of the receiving NI the credit counter will be increased with the amount of that data. Therefore the receiver NI should send the amount of freed space in its queue to the sending NI. In AEthereal, in order to prevent wasting bandwidth in the NoC, these credits

are piggy-backed in the response packet. The response packet are sent from the receiver NI to the sender NI. Therefore the credits reaches to the sending NI. When credits are received by the sender NI, it adds them to the space counter. The mechanism of end-to-end flow control is depicted in Figure 5.2.

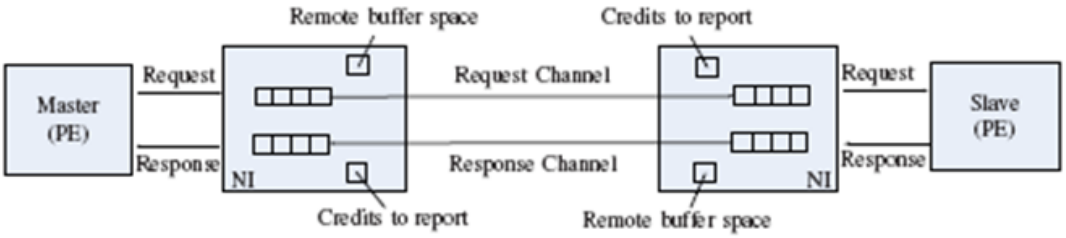


Figure 5.2: Credit-based end-to-end flow control in AEthereal [67]

Therefore another field of a header, is the credits that are piggy backed in the packet. In AEthereal the first *phit* of the header flit contains the header information. As depicted in the Figure 5.1.a, The first 27 bits of the header contains the path and the next 5 bits are the bits related to piggy-backed credits.

In AEthereal NoC, each *phit* is composed of 34 bits. The 2 most significant bits are the sideband bits that are added to the data words. Figure 5.1.b depicts the format of a flit. The side band bits, depending on which *phit* in the flit are they, provide us with different information about the packet and the corresponding flit. As shown in Figure 5.1.b, in the first *phit* flit the 2 side band bits define whether the flit is belongs to an empty packet or not. Moreover they define if the flit belongs to a packet that is a BE packet or a GT packet. Therefore there are 3 combinations defining three cases:

- **00:** If the packet is an empty packet.
- **01:** If the packet is a BE packet.
- **10:** If the packet is a GT packet.

The side band bits in the second *phit* of the flit show the number of *phits* in the flit that contain valid data [1]. For example the *phit* that contains the header information will not be counted as valid data. Therefore, as depicted in the Figure 5.1.b the side band bits of second *phit* in the header flit shows the number 2. This means that only the second and third *phit* in the header flit contains valid data.

The side band bits in the third and last *phit* of the flit, define if this flit is the last flit of the packet or not. In other words they define the end of the packet.

[1] Valid data means the data that belongs the message that should be transported between the source and destination.

5.1.2 Message format

Transactions between IP cores are the reads and writes that are done between IP cores. Each transaction is composed of one or more messages. There are two types of messages request message and response message [30]. For example, in a write transaction, the request message contains the data that should be written in the Destination IP core. Meanwhile, there maybe no response message for a write transaction or there may be a response message containing an acknowledgment.

Figure 5.3 depicts the format of a write message. As depicted in this figure, the first word contains control data and the second word contains address [30]. From third word until the last word contain the valid data that should be written in the destination IP core.

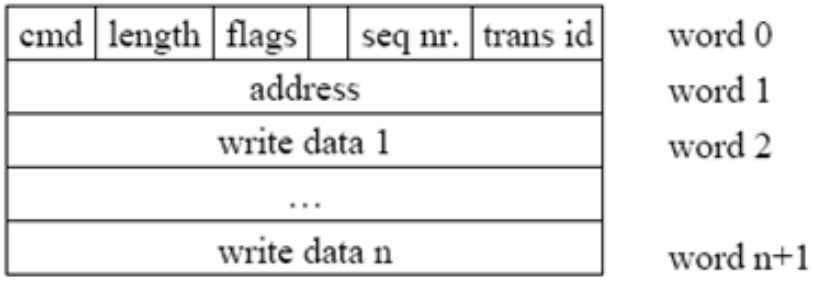


Figure 5.3: Message format in AEthereal [30]

5.1.3 Router interface

The router interface of the AEtheral NoC is composed of the input ports and output ports explained below: Input ports:

- **Clock.** The router works synchronously with the rest of the elements on the chip. Therefore *clock* signal of the router is derived from the clock signal of the chip.
- **Reset.** The *reset* signal of the router is also derived from the global reset of the chip.
- **Data-in.** *Data-in* has an equal width to the size of *phits*. The flits of packet is received *phit* by *phit* at this input port from the previous router in the path.
- **Flow-control-in.** This signal manages the link-level flow control between the router and the next router in the path. This signal is sent by the next router in the path to the router. Whenever this signal is active, it means that the next router has enough space to receive the next flit of the packet from the current router.

- **Scan-enable.** This signal defines whether the internal flip-flops of the router should be in normal mode or should be in scan-mode. This signal should be produced by a test infrastructure whenever there is a need to scan-in test stimuli to the scan-chained flip-flops of the router. Also when the test-response needs to be scanned-out from the scan-chained flip-flops, this signal should be active.
- **Test-reset.**

Output ports:

- **Data-out.** The *data-out* output port has the width equal to the *phit* size. It transfers the *phits* of the packet flits to the next router in the path.
- **Flow-control-out.** This signal is sent to the previous router in the path, in order to provide link-level flow control between the previous router in the path and the current router. If the current router has enough space to receive the next flit of the packet from the previous router in the path it activates this signal.

5.2 Wrapper design

For illustrating wrapper top-level design first, the wrapper interface design will be explained. Second, the internal architecture of the wrapper (the blocks that exist in the wrapper) will be introduced. At last the way the wrapper behaves and the data flow in different phases of test will be explained.

5.2.1 Inputs and outputs of the wrapper

The wrappers will be instantiated in the network. Since they cover the routers in the network, their interface should have all the ports of the routers. But in addition to the ports that are the same as router ports there are two more ports in the wrappers that are used to switching the router mode to test/functional mode. The wrapper ports that are similar with router ports are :

Input ports:

- **Clock.** Input *clock* signal of the wrapper is derived from the *clock* signal of the chip.
- **Reset.** Input *reset* signal of the wrapper is derived from the *reset* signal of the chip.
- **Data-in.** *Data-in* port of the wrapper has width equal to a *phit*. It receives the *phits* of the test stimuli packets from the last router in the request path (the path from TPG to the router-under-test).
- **Flow-control-in.** This signal, the same as *flow-control-in* signal of the router is used for link level flow control. In the router it is used for link level flow

control between two consequent routers in the path. But, here it is used for link level flow control between the wrapper [2] of router-under-test and the first router in the response path (the path from the router-under-test to the TRA). Whenever the the first router in the response path has space to receive three *phits* [3] of the test response packets, it notifies the wrapper by *flow-control-in*. This is depicted in Figure 5.4. In this figure, *RUT* is the router-under-test. In the wrapper is this figure *LNI* is the light NI that is one major block of the wrapper that takes care of test packets. Light NI is discussed in detail in next section. *Test-resp-pack* in the figure is the test response packet and *fc-in* is the flow-control-in.

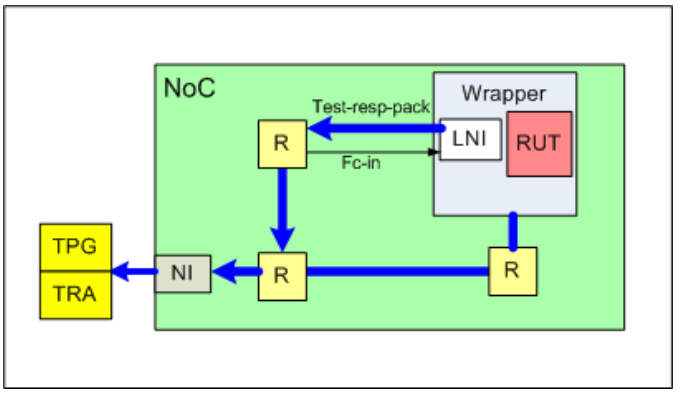


Figure 5.4: link-level flow control between wrapper and first router of response path

Output ports:

- **Data-out.** *Data-out* port of the wrapper has the width of a *phit*. It transports the *phits* of the test-response packets to the first router in the response path.
- **Flow-control-out.** This signal is also for link level flow control. It manages the link level flow control between the router-under-test and the last router in the request path. Whenever the wrapper has enough space to receive the next three *phit* (that becomes a flit) of the test stimuli packet from the last router in the request path, it sends an active *flow-control-out* signal to the last router in the request path. This is depicted in Figure 5.5. *Test-stimul-pack* in the figure is the test stimuli packet and *fc-out* is the flow-control-out.

The additional ports in the wrapper are:

- **Input** functional-test mode. *Functional-test* mode input signal of the wrapper defines if the router should be in functional mode or in test mode. In fact it defines if the wrapper should be transparent in functional mode or it should

[2] In fact between the light NI of the wrapper and next router. This is the light NI that has the test response packet and should send it to next router.

[3] The data unit transmission is flit in BE packets and each flit is three *phits*.

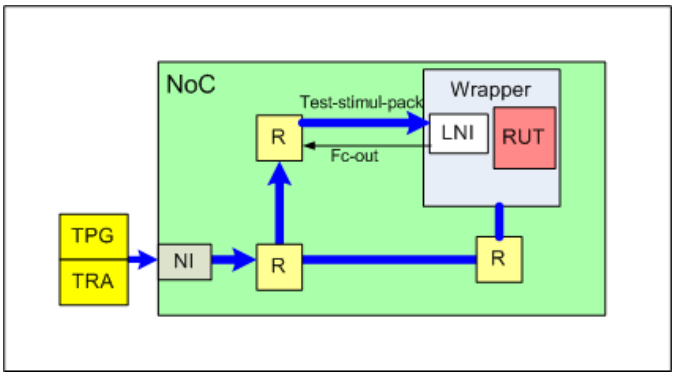


Figure 5.5: link-level flow control between wrapper and last router in request path

do some process on the packets in test mode. If the router should be in functional mode this signal has to be "0" and in test mode it should have the value "1". This signal is produced by the high-level scheduler for each of the wrappers around routers.

- **Output** ready. *Ready* signal is an output signal that is produced in the wrapper, whenever the router-under-test of the wrapper has been tested completely and the test procedure is finished. Moreover, in order to *ready* signal become activated, the wrapper should be sure that the connections between the attached NI to TPG-TRA (master NI) and light NI (slave NI) has been closed. The connection between the master NI and the light NI can be closed whenever the light NI receives the last credits of the end-to-end flow control (between master NI and slave NI) from the master NI. End-to-end flow control should exist between master NI and the light NI in order to avoid deadlock.[4]Each wrapper has one *ready* signal output that will be sent to the high-level scheduler.

5.2.2 General architecture

The routers in NoC normally are connected to the rest of the NoC by their functional ports. In the proposed method first the routers will be tested against the possible faults that may occur in the internal flip-flops. In this phase the router receive/send data from/to its scan pins. This data should be serial streaming data. Therefore the proposed wrapper should connect the scan pins of the router-under-test to the rest of the NoC. Moreover, in this phase it should first of all it should de-packetize the test packets and then it should serialize the parallel test data that is achieved from test packets. second, the router will be test against the faults that may happen in the combinational logic part. for this step again scan pins of the router first should used to receive the initialization test vector. Therefore again

[4]The detailed mechanism of closing a connection between the light NI and master NI is explained in section 5.3.4.3.

there is a need to serialize the parallel test data that is extracted from the test packets. Then the test data should be applied to one of the functional input ports. for this reason the only main issue is that to which functional input port of the router the test data should be transfered. In this step there is no need to serialize the test data.

Therefore the main responsibilities of the wrapper are: packetization and de-packetization, serial to parallel and parallel to serial conversion, keeping serial data streaming, and sending test data at each time to the correct test port which can be either the functional port of the router or the scan pins of the router.

Based on the above mentioned services that we expect the wrapper to provide us, the wrapper consists of a number of blocks:

- Light NI : This block is responsible for de-packetization of the test packets that reach to the wrapper to produces simple test messages. It is also used for packetization of the test responses messages. Another crucial service that the light NI should provide us with is the end-to-end flow control.
- Test block : The test block is the control block of the wrapper. Test block by activating proper control signals at the proper time, makes it possible to send/receive test data to/from correct ***test port*** of the router. Test block also controls that when the test data should be serialized and when it is sent as parallel test data. Also for the test responses. Test block also has the critical responsibility of halting the router and shift registers in the intervals between receiving test packets.
- Shift registers : Shift registers in fact can work as parallel to serial or serial to parallel converters. Moreover they can act as a simple register with parallel input and parallel output of the data.

Figure 5.6 shows the top level design of the wrapper. The light NI is the first receiver of the test packets. But as it is depicted in Figure 5.6, all packets arriving to the wrapper will be sent also to the light NI. But before they reach to the light NI, The two most significant bits of the *phits*, which in our case is 39 bits, will be *ANDed* with the signal *Functional/Test mode* that is one of the inputs of the wrapper. As a result of *ANDing* the two most significant bits of each *phit* with Function/Test mode signal, if the Function/Test mode signal is "0" (in functional mode), the two most significant bits of the *phits* becomes "00". If the router should be in test mode, the Function/Test mode signal is "1" and the result of AND gate will be the exact same two most significant bits of the *phits*, without any change. Therefore when the light NI receives the *phits* if they are "00", it will drop them as empty *phits*. This means that the light NI only processes the test packets and not the normal data packets that are being transfered between IP cores.

5.2.3 Wrapper behavior

For better understanding the responsibilities of each block of the wrapper, the flow of data in the wrapper in a complete test process (two phases of testing the router)

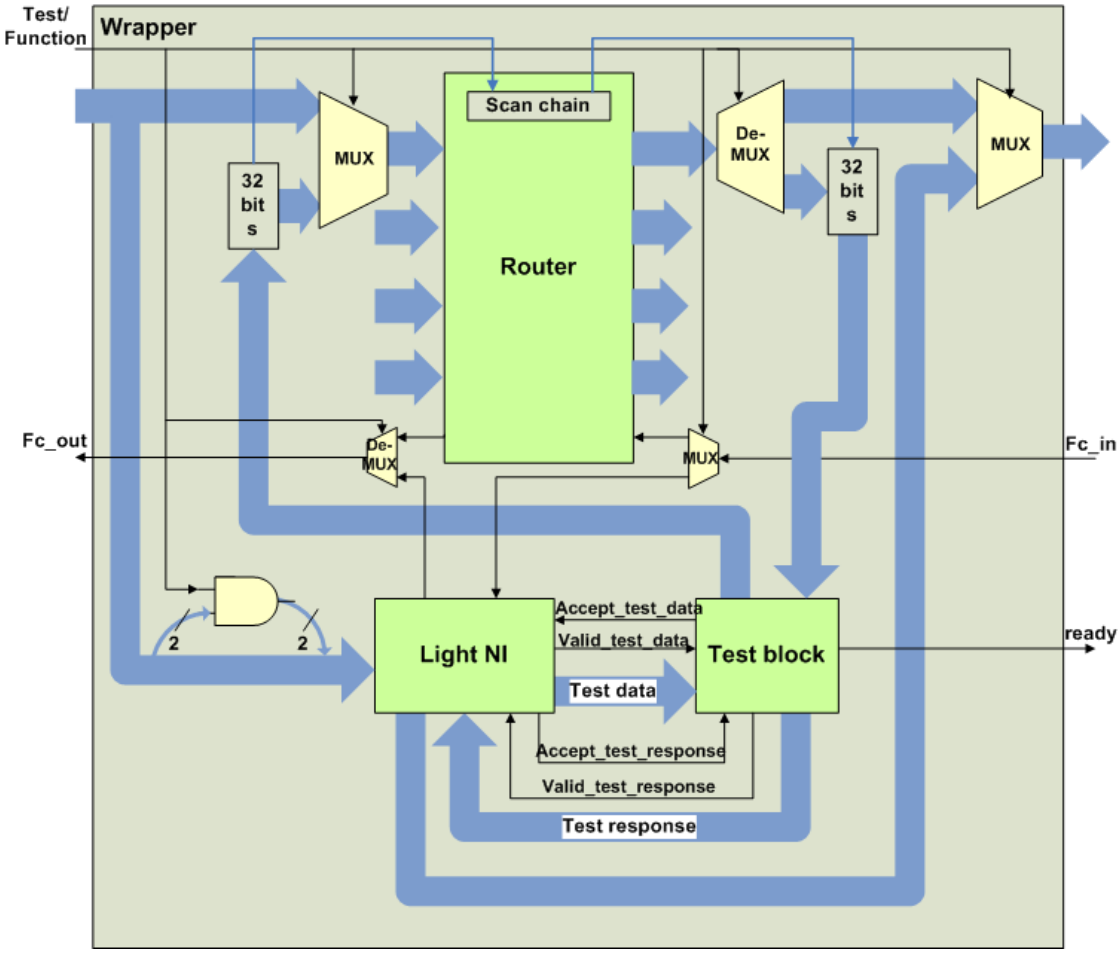


Figure 5.6: General wrapper architecture for testing NoC routers

will be explained below.

(a) **In first phase.** In first phase of test a test packet containing test pattern for testing the internal flip-flops should be sent to the router-under-test. When the test packet reaches to the wrapper (that wraps around the corresponding router-under-test), first it should get de-packetized by the light NI. The light NI receives the test packet and produces test message in the form of pure message accompanied by a valid and accept signals. The test message will be sent to the test block. The first valid signal of the test message that is received by the test block will turn on the test block. Test block when receives the **first** valid test message signal understands that the test message contains test pattern for testing the internal flip-flops. Therefore it activates first the control signals related to parallel to serial conversion and sending test data to the scan-in pins of the router and then control signals related to receiving test response from scan-out pins and serial to parallel conversion. Therefore

when the test responses that are the result of testing internal flip-flops of the router are scanned out they will convert to parallel data forming words. The test response words then will be sent to the light NI to get packetized. The light NI packetize the test responses and insert the return path to the TRA in the header of the test response packets. The test response packets will be the output packets of the wrapper.

(b) **In second phase.** In the second phase that is testing the combinational logic part of the router, as explained in previous chapter, first the test pattern for initializing the router state should be inserted to the scan chained flip-flops of the router. When the test packet containing initialization pattern of the internal flip-flops arrive to the wrapper again first it get de-packetized by the light NI. The obtained test message will be sent to the test block. Test block at this time knows that the first phase of testing is finished and at this step the test message that is received needs to initialize the router state. Therefore at this step the test block only activates the control signals for parallel to serial conversion and scanning in the test data. After this the test block is waiting for receiving the next test data.

Next test packet that is received like all previous ones will get depacketized by light NI and sent to test block, at this point when test block receives a test data, it knows that this test data is for stimulating the combinational logic part of the router, and it should be sent directly to one of the functional ports of the router. Therefore it only activates the signal for parallel loading of the test data to the functional input port of the router.

After one clock cycle that is the the time that test stimuli has been applied to the combinational logic of the router, the response of testing combinational logic part will be ready on one of the functional output ports. It is time to receive the test responses. The test response of combinational logic part test is composed of two parts:

i. The output of the combinational logic part after test. This is the data that is produced at the output port of the router after test application.

ii. The state of the router after test application. The state of the router is defined by the contents of the internal flip-flops of the router.

Therefore, the test block should first activate the control signals related to load out the test response from one of the functional output ports of the router. Afterward, it has to activate control signals for scanning-out the contents of the internal flip-flops of the router, convert them to parallel data words and send to the light NI. The light NI packetizes the test responses and sends them to TRA to get analyzed.

Figure 5.7. shows the sequence of works to be done in phase 1 and phase 2. As depicted in this figure, the sequence of works in phase 1 and phase 2 can be overlapped in order to decrease the test time. In other words when at first phase the test block is scanning-out the test response of testing the internal flip-flops, it can start scanning-in the initialization vector in second phase of test.

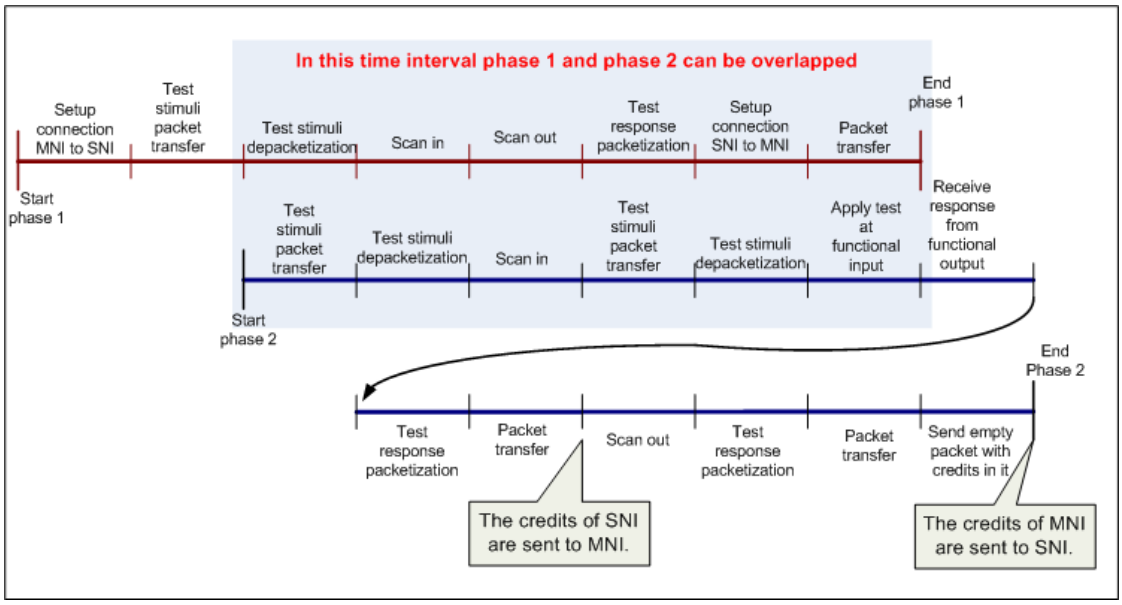


Figure 5.7: Sequence of actions in phase 1 and 2 of test

5.3 The light NI

The light NI in each wrapper is receiver of the test stimuli packet from the NI attached to TPG. After receiving the test stimuli packet, it de-packetize it to test stimuli message. The message will be sent to the router-under-test. On the other hand the light NI is the sender of the test response packets. First it will receive the test response message. Then it convert the message to packet and sends it to NI attached to TRA.

For receiving the test stimuli packet to the router-under-test and sending the test response packet from the router-under-test, there should be connections between the attached NI to the TPG-TRA (master NI) and the corresponding light NI of router-under-test (slave NI). This connections are depicted in Figure 5.8. The attached NI to TPG-TRA block is the master NI and the corresponding light NI to the router-under-test is the slave NI. AS depicted in Figure 5.8., the request connection is from the master NI to the slave NI and the response connection is from slave NI to the master NI. In this figure, *MSB* stands for Master Sender Buffer, *MRB* stands for Master Receiver Buffer, *SSB* stands for Slave Sender Buffer and *SRB* stands for Slave Receiver Buffer.

The light NI has some of the functionalities that a normal NI kernel should provide us. Therefore, almost the same architecture of the normal NI kernel is used for the light NI. However the light NI is different in number of issues. Due to this reason, some parts of the normal NI architecture should be modified. In what follows these differences are explained. Also the solution for them and the way the light NI should be modified is given.

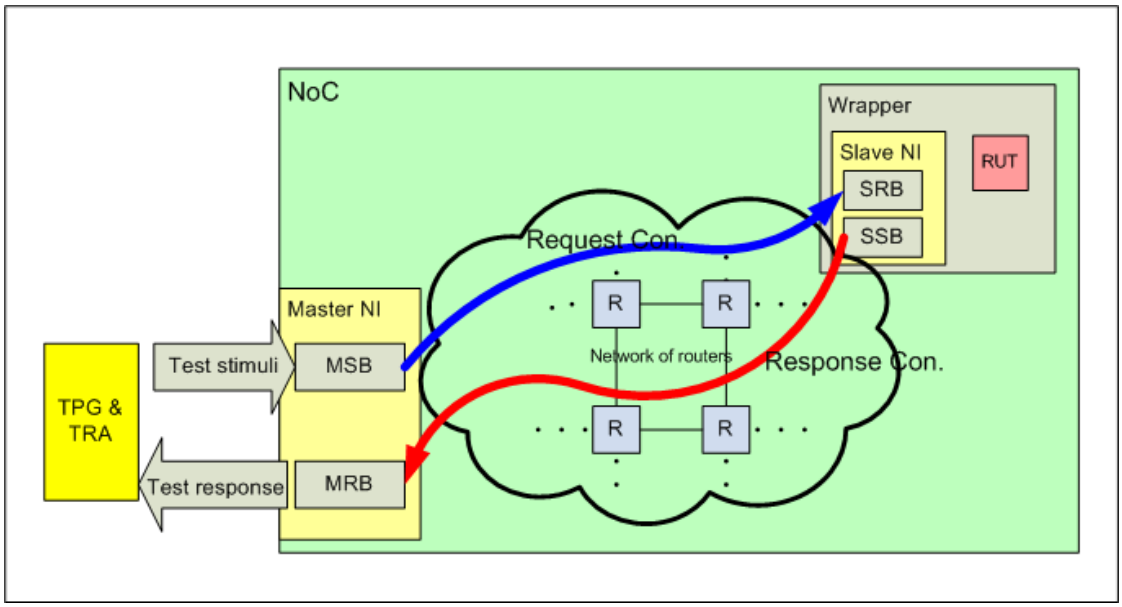


Figure 5.8: Request and response connections between NIs for test data transportation

5.3.1 Non-programmability of the light NI

Since the light NI is being used in the wrappers, and wrappers are around routers, if the router-under-test is a non-boundary router, The light NI in the corresponding wrapper is not accessible via the NI shells. Therefore they can not directly receive the DTL message that programs the NI kernel and therefore, can not be programmed directly. Another alternative is to program the light NI via the master NI. But in this way, there is a need to first send a packet from the NI attached to the TPG-TRA (master NI) to the light NI.

Due to two reasons it is decided to eliminated the programmability of the light NI. First, in this test methodology it has been tried to lower the related traffic of the test packets as much as possible. This is because we do not want that the testing affects the normal functionality of the NoC which is transportation of normal packets between IP cores. Second, in the added wrappers around the routers we need to have as small hardware overhead as possible. Therefore, it is decided that the light NIs inside the wrappers are not programmable. But there are a number of issues that should be solved in non-programmable light NIs.

With programming NI kernel, services that are explained below are being addressed:

- Initializing space control with the amount of space available in the remote NI (receiving NI) for end-to-end flow control.
- Giving the path to the destination that should be added to the header of the packet.

- Defining the value of some of the control signals in the NI that are used to define if a packet is BE or GT, if there should exist end-to-end flow control and if the queue in the NI should be activated or not.

With eliminating the programmability of the NI, we should add some mechanisms to provide us with the above mentioned services that previously were provided by programming the NI. Initializing the space control will be discussed in section 5.3.4 and calculation of the path to destination in light NI which is the return path to from light NI to the TRA will be discussed in section 5.3.3. For giving value to the control signals, by looking at the functionality that we expect from the light NI we can give constant values to the control signals. The control signals that are tied in to a constant value are:

- **Gt-valid.** This signal is produced by GT-scheduler and acts as a flag to show that the current packet that is forming in the NI is of type GT. This signal is also as an input to the BE-scheduler. This is in because in AEthereal, GT packets have priority over BE packets. Therefore BE-scheduler should know if there is an GT-packet currently waiting to be sent in the NI or not. If there is, the BE-scheduler should schedule the BE-packets after the GT packet is sent and there is no GT-packet in the NI waiting to be sent. Since in the light NI, there is no need for GT packets and therefore no need for GT-scheduler, This signal is tied in to '0'. By making this signal '0', the BE-scheduler understands that there is no GT-packet in the light NI and schedules the BE test packets on the correct time.
- **Flow-control-up.** This signal enables end-to-end flow control in the request path . In the light NI, this signal is tied in to '1' which enables the end-to-end flow control in the request path[5].
- **Flow-control-down.** This signal is the same as Flow-control-up, but for response path. In the light NI, this signal is also tied in to '1'. This enables end-to-end flow control in the response path.
- **Enable-queue.** *Enable-queue* signal is responsible for enabling one of the queues of in the NI that its stored data has to be scheduled and sent out as a packet. In the light NI, there is only one queue and it should always be enabled. This is because there is no other queue that needs to be activated, therefore we let the only queue of the light NI be always enabled. This is the reason this signal is tied in to '1' in the light NI.

Hence, in the light NI, all the infrastructure for programming the NI can be eliminated. This includes the block kernel-reg-ctrl, the register-files that are: fc-up-reg-file, fc-down-reg-file, enable-queue-reg-file and gt-be-reg-file.

[5]The reason there is a need for end-to-end flow control will be discussed in section 5.3.5

5.3.2 Only BE connections in light NI

As explained in previous chapter, the test packets are being sent using BE connections. In the NIs that the test packets are formed (The attached NI to TPG and the light NI), the packets are formed as BE packets. This is done by setting up a flag in the packet The 2 side band bits of the first *phit* in the flit defines that packet is BE type or not..

The NIs responsible for making test packets are the NI attached to the TPG-TRA and in the light NI in the wrappers. Therefore in these NIs we only need the infrastructure for producing BE packets. The NI that is attached to the TPG-TRA is like other normal NIs (consisting NI shell and NI kernel) and there is no need to modify it. But, since we need to reduce the hardware overhead of the wrappers around routers, the unnecessary blocks in the light NI (GT constructs) should be deleted. In the NI kernel architecture of AEthereal some of the blocks are unnecessary and should be removed for making light NI. These blocks are:

- **GT scheduler.** This block is responsible for scheduling the GT packets. It schedules the GT packets based on the reserves time slots for each packet. The reserved time slots for packets are stored in the *slot table*. Since the test packets are not sent as GT packets, there is no need to have a GT scheduler in the light NI.
- **Slot table.** Slot table is used to provide Time Division Circuit Switching routing for the GT packets. Therefore, when test packets are not sent by GT connections, there is no need for slot table.

Moreover, since there is no GT blocks in the light NI, multiplexers that were used for multiplexing between GT control data and BE control data can be eliminated.

5.3.3 Calculation of the return path

The path from source to destination in normal NI kernel is given by the DTL data that configures the NI. However, the light NI is not programmable. Therefore it can not be programmed and receive the return path [6] by the DTL control data. This is the reason that there is a need for a mechanism to calculate the return path in the light NI.

As it was discussed in previous chapter the TPG and TRA are both connected to the same port of the master NI. Figure 5.9 shows the way TPG and TRA are connected to NoC.

As it was depicted in Figure 5.8, in the request path the source is the TPG-TRA block and the destination is the router-under-test. In the response path the source is the router-under-test and the destination is the TPG-TRA block. It is obvious that the only difference in request path and response path is that the order of

[6]The return path is from the slave NI in the wrapper of the router-under-test to the master NI attached to TPG-TRA.

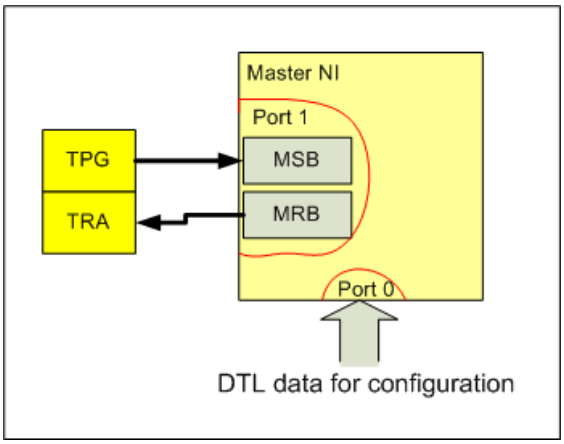


Figure 5.9: Connection of TPG-TRA to master NI port

routers in the path from source to destination is reversed. This means that the request path is the bit reverse of the response path.

When a test packet containing test stimuli arrives to the wrapper, the request path that is stored in the header of the packet can be extracted. The light NI by bit-reversing this extracted path can calculate the response path. After calculation of the response path by the light NI, it will be added in the header of the test response packet.

But there is still a problem. The problem is that when the master NI is forming the request path [7], it does not insert its own $Queue_{id}$ [8] that the test stimuli packet reside in its input queue. This is important because TPG and TRA are connected to the same port number of the NI. Therefore the test response packet should also be received in the same queue of the NI that the test stimuli packet were stored. The difference is that test stimuli packet that should leave the NI are stored in the input queue and the test response packet is stored the output queue of a specific port of the NI. Therefore in the return path, the $Queue_{id}$ that the test response packets should enter can not be known by bit-reversing the request path. In fact by bit-reversing the request path only the return path through router network is defined. As a result, the test response packet when arriving to the master NI, does not know in the output queue of which port it should be stored.

For solving this problem we have to add one more step to calculate the return path [9].

[7] Master NI in request path is the sending NI.

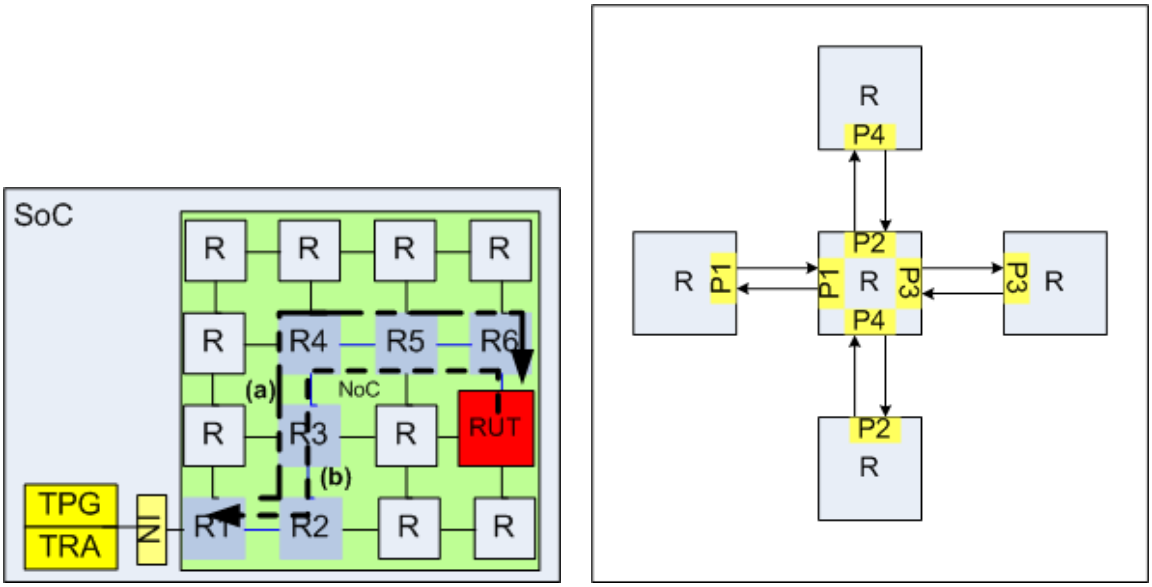
[8] Since the TPG-TRA block is connected to port 1 of the master NI, the $Queue_{id}$ that should be inserted in the path is '00001'.

[9] In this project the header format in first implementation of AEthereal is considered. In this format the number of bits for $Queue_{id}$ selection is fixed. The first 22 bits of the header *phit* gives us the hops through router network. The next 5 bits define the port of the receiving NI that the packet will be stored in its queue.

- First, the bit-reverse of first 22 bits of the path in header should be calculated. The mechanism for bit-reversing will be explained in what follows. The bit-reverse of first 22 bits of the path gives us the return path through the router network.
- Second, 5 bits that define the $Queue_{id}$ of the receiving NI [10] will be concatenated to the end of reversed router hops that is obtained from the first step. This 5 bits selects the queue that the data of the test response packet should be stored in it.

In this project the fixed value for $Queue_{id}$ is considered to be "00001". This means that the data of test response packets will be stored in the input queue of port number 1 of the NI. Therefore the TPG-TRA should be connected to port number 1 of the NI. TPG will send data to input queue of port number 1 and TRA will receive data from output queue of port number 1.

Bit-reversing mechanism is shown in an illustrative example in what follows. For illustrating this example an alternative request and response path are chosen that is depicted in Figure 5.10.a. In this figure, it is assumed that the request path from TPG-TRA to the router-under-test is the path shown by line a. The response path from router-under-test to TPG-TRA is shown by line b. The paths are chosen just with the purpose to give a comprehensive illustration on bit reversing and return path calculation.



(a) Path to router-under-test

(b) Connection of router ports

Figure 5.10: Path to the router-under-test and routers connections

[10] Master NI

Figure 5.10.b is showing the router's connections to neighboring routers in a NoC with a regular mesh-based topology such as AEtheral. Figure 5.10.b shows the number of router's ports and the way they are connected to each other in a mesh-based topology. In this figure the left port of the router is assumed to be port number 1, the up port is assumed to be port number 2, the right port to be port number 3 and the down port to be port number 4.

Each port of the router is connected to one and only one other router. Moreover, it is exactly defined that each port of the router is connected to which other port in the next or previous router. For example, in regular mesh-based topology shown in Figure 5.10.b, the pair of ports that are connected together and the way they are connected is illustrated in Table 5.1:

	input-output port connections	
$\text{Port}_1 \rightleftharpoons Port_3$	$\text{output}_1 \rightarrow input_3$	$\text{output}_3 \rightarrow input_1$
$\text{Port}_2 \rightleftharpoons Port_4$	$\text{output}_2 \rightarrow input_4$	$\text{output}_4 \rightarrow input_2$

Table 5.1: Pair of connecting ports in a regular mesh-based topology

Based on the Table 5.1, we can define a definition for **router port's complement**.

Definition: Each port of a router has a complement port. The complement of each port of router is the one that the router port is connected to it in the router network.

Therefore, the router port's complement in AEthereal can be defined by looking at Table 5.1. The complement of input ports is an output port. This is because the input ports are connected only to an output port. The complement of output ports is an input port, for the same reason.

In what follows first the way a packet will traverse the request path from source to destination will be illustrated by the example shown in Figure 5.10.a. Second, calculation of the response path in the light NI in two steps (by bit-reversing the request path and concatenating $Queue_{id}$) is explained. Finally, the way test response packet will traverse the response path will be demonstrated.

5.3.3.1 Traversing request path

In Figure 5.10.a, the request path from TPG-TRA to the router-under-test will be defined by router hops plus the $Queue_{id}$ of the light NI in the wrapper around the router-under-test. Therefore, the request path will be:

$$Request\,path \;=\; o_3\; o_2\; o_2\; o_3\; o_3\; o_4\; Queue_{id}$$

The test packet will traverse this path until it reaches to the destination. The final destination is the corresponding output queue of the light NI that its $Queue_{id}$ is at the end of the path. Below it is explained how the packet reaches to destination by traversing this path.

In the request path depicted in Figure 5.10.a, when a test packet is sent to $router_1$ in the path by the master NI, as the first hop in the path shows, it will be routed to $output_3$ of the $router_1$. As presented in Table ... the $output_3$ is connected to the $input_1$ of the next router. Therefore, the test packet will be received from $input_1$ of $router_2$ in the path. In the $router_2$, by looking at the second hop in the path, the test packet will be routed to $output_2$. Again as shown in Table ..., $output_2$ is connected to $input_4$ of the next router. Therefore the packet will be received from $input_4$ of $router_3$ in the path. In the same way, the test packet traverses hop_3, hop_4, hop_5 and hop_6 until it reaches to $router_6$ in the path. At this point, the next hop (which is the last one) defines to which output queue of the receiving NI the packet should go. The light NI is the receiving NI and in light NI we have only one input queue. Therefore the $Queue_{id}$ in the request path is "00001" and the test packet enters the $output-queue_1$ of the light NI.

5.3.3.2 Calculating the return path

In the response path the light NI should build the return path. The return path in the light NI will be built by in two steps:

(a) **Reversing the router-hops.** For reversing the router hops only the router-hops [11] in the request path will simply be reversed. In the above example we have:

$$Router\ hops\ of\ Request\ path = o_3\ o_2\ o_2\ o_3\ o_3\ o_4$$

For reversing the hops, the steps given below should be followed:

i. In the beginning, instead of output ports (that are given as hops in the request path), the complement of corresponding input ports will be replaced in the path. Therefore when the complement of corresponding input ports are replaced in the request path instead of output ports for the hops, we have:

$$Replaced\ hops\ in\ Request\ path = \bar{i_3}\ \bar{i_2}\ \bar{i_2}\ \bar{i_3}\ \bar{i_3}\ \bar{i_4}$$

ii. In this step, all the hops in the path from previous step will be reversed. The result in the above example is:

$$\Rightarrow \bar{i_4}\ \bar{i_3}\ \bar{i_3}\ \bar{i_2}\ \bar{i_2}\ \bar{i_3}$$

iii. Based on the complement ports [12], presented in Table 5.1, the complements are calculate and replaced in the path, as a result we have:

$$\Rightarrow Reversed\ router\ hops = o_2\ o_1\ o_1\ o_4\ o_4\ o_1$$

[11] The first 22 bits.

[12] The complement of an input port, is the output port that is connected to it in the NoC topology.

The mechanism for reversing router hops shown in the above example can be generalized for a path with N routers, as below:

Given router hops : $h_0 \ \ldots \ h_{n-1}$

Reversed router hops: $\bar{h_{n-1}} \ \ldots \ \bar{h_0}$

(b) **Concatenating $Queue_{id}$ of the receiving NI.** In this step, a fixed $Queue_{id}$ with a known number of bits will be concatenated to the end of reversed bits. The fixed $Queue_{id}$ is chosen to be "00001". This is because the TPG-TRA core is connected to $port_1$ of the NI. Therefore, in the above example the final return path will be:

$$o_2 \ o_1 \ o_1 \ o_4 \ o_4 \ o_1 \ Queue_{id}$$

In general the final request path will be:

$$\bar{h_{n-1}} \ \ldots \ \bar{h_0} \ Queue_{id}$$

5.3.3.3 Traversing the return path

Test response packet will traverse the response path to reach to the TPG-TRA. In the response path given above, the first hop is 0_2 that is connected to $input_4$ of the $router_5$. Therefore test response packet will be received via $input_4$ of $router_5$. The rest of the hops until sixth hop will be traversed the same until the packet reaches to the $router_1$. At this point the next hop is the $Queue_{id}$ in the master NI. Therefore the test response packets will be stored in the output queue of the master NI.

5.3.4 End-to-End flow control in the light NI

In order to test a router, two connections should be made; request connection and response connection. Request connection is from the master NI [13] to the Slave NI The light NI in the wrapper around the router-under-test. The response connection is from the light NI to Master NI.

TPG produces streaming test stimuli and feed it to the request connection. This means that TPG is producing one bit of test stimuli in each clock cycle. TRA also consume and analyze the test responses in a streaming fashion. Thus, it analyzes one bit of test response in each clock cycle. Similar to test generation and analysis, the test application to the router-under-test is in streaming fashion. In other words, during test application one bit of test stimuli is applied to the router-under-test in each cycle and after specific time, one bit of test response will be produced at each cycle. Therefore, if the transportation of test data to/from the router-under-test is also in streaming fashion, there is no need for end-to-end flow control. This is because TPG and TRA are working with the same speed.

However, test stimuli and test response transportation can not be in streaming fashion. Since the test packets are sent by BE connections, there is no guarantee

[13] The attached NI to TPG-TRA

in the arrival time of the test packets to the destination. Since the test data [14]transportation may not be in streaming fashion, there is a need for end-to-end flow control in both connections; request and response. In what follows, the reason that we need end-to-end flow control in each of the connections is explained in more detail. But first an important remark is given about the data flow of BE packets in AEthereal.

Remark: In AEtheral NoC, the BE packets are routed based on wormhole flow control. As explained in chapter 2, in wormhole flow control flits are the units of data transmission. Since the test packets are treated as BE packets, they are transfered flit by flit. Therefore, in each transmission, there should be enough space in destination for accepting one flit.

5.3.4.1 The reason for end-to-end flow control in request connection

The need for end-to-end flow control is demonstrated by an example. In this example, it is assumed that the test stimuli is large enough that it can be divided into number of flits. After receiving number of first flits, if the test responses are not transfered on-time out of the light NI, the output queue of light NI, input register, scan chain of the router-under-test and the output register become full. In this situation master NI should wait until one flit of the test response [15] can be delivered out of the light NI. As a result, the output register will send its contents to the light NI, scan chain will scan-out its contents to the output register, input register will scan-in its contents to the scan chain and light NI can send new test stimuli messages to the input register. After three cycles of this process three words will be loaded out of the light NI. Therefore there will be one flit space in the light NI to receive the waiting test stimuli flit in the master NI.

The above example shows the reason we need to control the amount of available space in the corresponding light NI to the router-under-test. In other words, this is the reason that there should exist end-to-end flow control in the request connections from the master NI [16] to the slave NI [17].

5.3.4.2 The reason for end-to-end flow control in response connection

Each of the light NIs before sending the test response flits to the master NI [18], should check the amount of space available in the attached NI to TPG-TRA. If there is enough space in the master NI to accept one flit of data, the corresponding light NI will send the flit. One may think that since TRA is consuming test responses in a streaming way, there is no need for controlling the amount of available space in the master NI. But, even by considering one router-under-test at each

[14]Test data means test stimuli or test response.
[15]One flit of test response is equal to three *phits* of test reponse.
[16]NI attached to the TPG-TRA
[17]Corresponding light NIs to router-under-test.
[18]attached NI to TPG-TRA

time, we still need end-to-end flow control for response connections. Below there is an example to show this fact.

When a flit of test response packet reaches to the master NI, after de-packetization [19] the words of the message will be stored in the output queue of the NI. TRA will receive the test response message word by word from the output queue. TRA consumes at each clock cycle one bit of the test response word. This means in order to consume a 32 bit word test response Stored in the output queue of the attached NI it needs 32 clock cycles. Therefore, in order to make a free space for accepting a new flit in the output queue of the NI, TRA needs $32 \times 3 = 96$ clock cycles. The de-packetization process in the attached NI to TPG-TRA also needs number of clock cycles. But if the de-packetization process is done in a shorter time than analyzing three words [20] of test response that are stored the output queue, the new test response flit to be stored in the output queue will be ready earlier than a space for accepting a flit in output queue becomes free. This issue can be illustrated with an example.

Assume each entry in the output queue has the capacity to store a word with 32 bits. The output queue is considered to have 9 entries (equal to 3 flits). In the NI, extracting a word of test response message out of a *phit* of flit of a test response packet process [21] is assumed to take 5 clock cycles. it is assumed that the length of a test response from a router-under-test is 15 words (equal to 5 flits). When the first word is received and stored in the output queue, the TRA can start analyzing the test response by receiving one bit of the test response word at each clock cycle. Therefore after 32 clock cycles the first entry of the output queue will be free again. But, before this happens two other words can be received and stored in the two other free entries of the output queue in 10 clock cycles. At this point, there is still 22 clock cycles to pass in oder to first entity of output queue become free again. Therefore one complete flit of the output port is occupied. After passing 30 more clock cycles, the remaining 6 more entries of the output queue will be also full. At this point the first entity of the output queue after 22 clock cycles has been emptied again, but still there is no space to receive a complete flit. Therefore if at this point the 4^{th} flit of test response from the light NI arrives to the master NI, there is no space in the output queue to store it. The light NI should wait for 56 $32 \times 3 = 96$ clock cycles it last until the first flit in the output queue becomes free. After $8 \times 5 = 40$ clock cycles the 8 remainder entities of the output queue become full. Therefore the 4^th flit can be received after $96 - 40 = 56$. more clock cycles until the first flit in the queue becomes available. This example shows us that the light NI of a router-under-test should have knowledge about the amount of space that is available in the master NI.

[19] extracting test message words out of the *phits* in the flit
[20] One flit has three words(*phits*)
[21] De-packetization process

5.3.4.3 End-to-end flow control implementation

As mentioned in previous section, end-to-end flow control is realized by two counters; space counter in the sender NI and credit counter in the receiver NI. Space counter in each NI, has the amount of free space in the receiver NI [22]. In other words the space counter in the sender NI defines that how many flits the sender NI can send to the receiver NI. The credit counter in each NI defines how many flits have leaved the NI, therefore how much space has been freed. The credits of a receiving NI are piggy backed in the response packet and sent to the sending NI. In sender NI, the received credits from the receiver NI will be added to space counter to calculate the new amount of available space in the receiver NI.

In the beginning, when a connection has to be established between sender NI and receiver NI, the space counter in the sender NI should be initialized with the amount of queue capacity in the receiver NI. This is defined by the number of flits that can fit in the queue. Initialization of the space counter is done by programming the NI.

At the end, when a connection has to be teared down between the two NIs, first all credits of the receiver NI have to be received by the sender NI. This is because after closing the connection, the space counter in the sending NI should have the correct value of the amount of available space in the receiving NI. Otherwise the next connection that will be established between the same sender NI and receiver NI will be defected.

Now we consider the above mentioned points in end-to-end flow control in request and response connections between attached NI to TPG-TRA (master NI) and a light NI (slave NI).

(a) **Request connection.** In the request connection master NI is the sender NI and slave NI is the receiver NI. Here, end-to-end flow control can be easily realized. For this reason, first by programming the master NI, the space counter in master NI will be initialized to a value equal to the capacity of the queue in light NIs. The credits of the corresponding light NI is piggy backed in the test response packets and is received by the master NI. Since there is always a test response packet in order to respond to the test stimuli packet, we are always sure that at the end the credits of the light NI will be received by the master NI. This assures us that the request connection will be always teared down at the end.

(b) **Response connection.** In response connection slave NI is the sender NI and master NI is the receiver NI. In response connection, end-to-end flow control can not be achieved as easy as in request connection. This is because of two main reasons.

- First, the light NI is not programmable. Therefore, we can not initialize the space counter in the light NI by programming it.

[22] Remote NI.

- Second, the last credits of the master NI (that here is the receiver NI) may not be received by the light NI (that here is the sender NI). In order to send the credits of the master NI to the light NI, they have to be piggy-backed in test stimuli packets. Therefore, when the packets of test stimuli are finished and there is no test stimuli packets to be sent to the light NI, the credits can not be transferred to the light NI of router-under-test. In this way the corresponding light NI does not have the correct information about the amount of available space in the master NI.

The two issues related to end-to-end flow control in response connection can be fixed as follow.

- To fix the first problem, the space counter in the Light NI can be initialized at reset time. At reset time, we can initialize the space counter to a constant value *M*. The queue capacity in the master NI should be equal or greater than *M*. For this reason the space-table-ctrl module in the NI should be modified as it is shown in Figure 5.11.

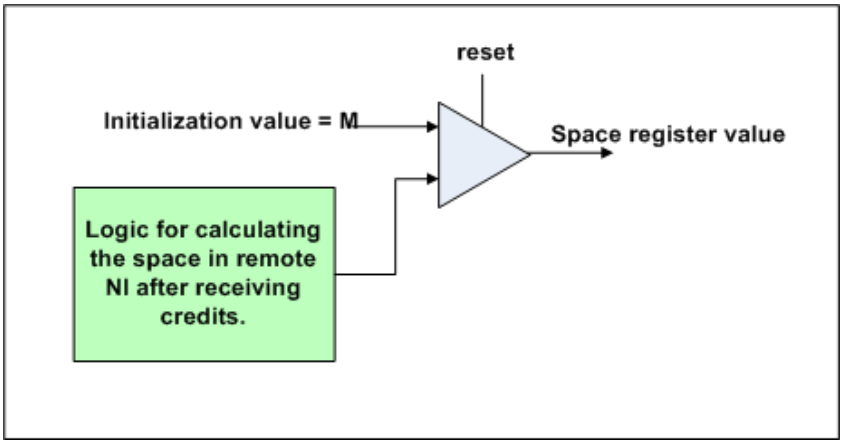


Figure 5.11: Initialization of space counter in light NI

- To fix the second problem, we should make sure that the credits in the attached NI to TPG-TRA will be transfered to the light NI. This can be done by sending an empty packet from the attached NI to TPG-TRA to the light NI that only contains a header with piggy-backed credits. In other words a packet that has only a header will be sent from master NI to the light NI. In this way, the last credits of the master NI can be received by the light NI. Therefore, response connection can be closed.

Due to above mentioned, in light NI after adding the received credits to the value of space counter, it should compare the value of space counter with the maximum value of space counter [23]. If space counter becomes equal to the maximum value, then there is a chance that the router can go out of test mode. But this can be decided only by the test block.

[23]The value that space counter was initialized to. This value is equal to M as discussed in section ...

The test block that knows when the whole test process is finished and the test process is finished after testing the internal flip-flops and the combinational logic part. Therefore, when current space value becomes equal to the initial space value, the light NI sends a signal may-ready to the test block. Then, in test block if it is the end of test process and the may-ready signal is *High*, the test block makes the signal ready "1" that is the output of the wrapper. The signal *ready* of each wrapper around a router, defines if the wrapper test is finished or not. If the test is finished, its corresponding *ready* signal becomes one. All of the *ready* signals corresponding to wrappers will form a vector. The vector size is equal to the number of routers in NoC and will be as input to the high-level scheduler. A router can go out of test mode by the high-level scheduler only when its *ready* signal is "1". Otherwise if the high-level scheduler changes the test mode of a router, before its corresponding light NI receives all the credits, the light NI can not receive the final credits from the master NI. Therefore, the space counter in the light NI will have an in-valid value.

5.4 The test block

From the overall functioning of the wrapper explained in previous section, one can understand that the test block is responsible for producing control signals. More over test block has the responsibility for timing the wrapper. In other words, test block should control at each time what has to be done. Figure 5.12. shows the top level design of the test block. As it is depicted in the figure, there is a state register that defines the state of the test block. First the interface of the test block (input/output ports) to the rest of the block in the wrapper will be explained. Second, The test block behavior in different steps of testing is illustrated and at the end the test block implementation is given.

5.4.1 Test block interface

The inputs to the test block are:

- **Clock.** The input *clock* signal of the test block is derived from the *clock* signal of the wrapper.
- **Reset.** The input *reset* signal of the test block is derived from the *reset* signal of the wrapper.
- **Valid-test-data.** This signal is sent from light NI to the test block in order to notify the test block that a test stimuli message is ready to be sent to the test block. Whenever this input signal is high the test block understands that there is a test stimuli word ready to be sent by the light NI to test block.
- **Accept-test-response.** This signal is sent from the light NI to the test block to inform the test block that the test response has been received and accepted by the light NI.

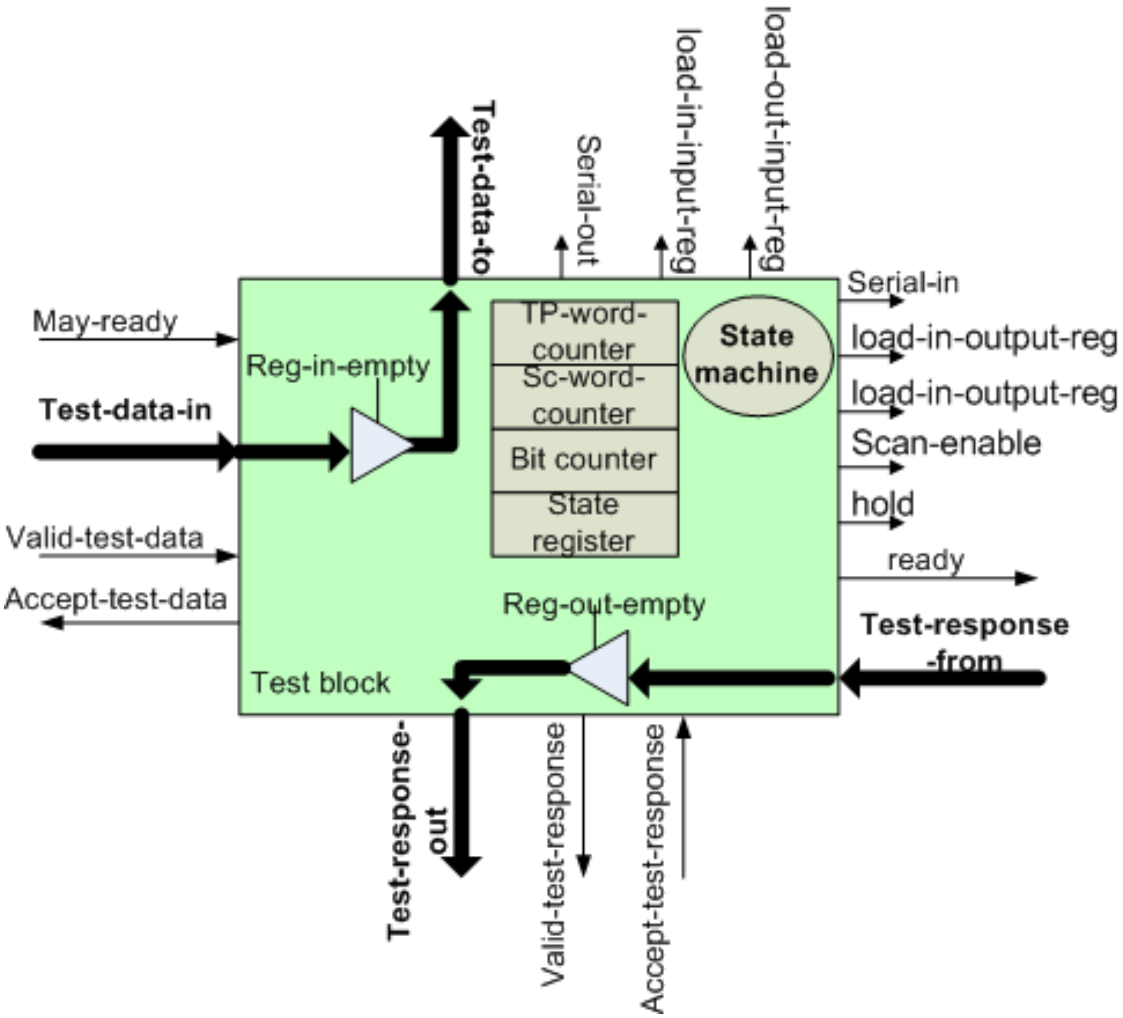


Figure 5.12: The test block

- **May-ready.** This signal is sent by light NI to the test block whenever the light NI receives an empty packet that contains only credits.[24]
- **Test-data-in.** Test-data-in is a bus with width equal to a word. It sends the words of test stimuli message from the light NI to the test block.
- **Test-response-from.** Test-response-from is also a word wide bus. It transfers the test response words from the output register to the test block.

The outputs from the test block are:

- **Accept-test-data.** This is a control signal and active value of this signal is "1" and in-active value is "0".

[24]This is a sign that maybe the test process is finished.

- **Valid-test-response.** This is a control signal and active value of this signal is "1" and in-active value is "0".
- **Serial-out.** This is a control signal and active value of this signal is "1" and in-active value is "0".
- **Serial-in.** This is a control signal and active value of this signal is "1" and in-active value is "0".
- **Parallel-load-in-input-reg.** This is a control signal and active value of this signal is "1" and in-active value is "0".
- **Parallel-load-out-input-reg.** This is a control signal and active value of this signal is "1" and in-active value is "0".
- **Parallel-load-in-output-reg.** This is a control signal and active value of this signal is "1" and in-active value is "0".
- **Parallel-load-out-output-reg.** This is a control signal and active value of this signal is "1" and in-active value is "0".
- **Scan-enable.** This is a control signal and active value of this signal is "1" and in-active value is "0".
- **Hold.** This is a control signal and active value of this signal is "0" and in-active value is "1".
- **Ready.** This signal is active when it is "1" and inactive when it is "0". When this signal is active, it shows that the test process of the router is finished. This signal will be sent to the hight-level scheduler.
- **Test-response-out.** It is a word wide bus that transfers test response words from the test block to the light NI.
- **Test-data-to.** This is also a word wide bus that transfers the words of the test stimuli from the test block to the input register.

5.4.2 Test block behavior

Test block control the timing and activity in the wrapper by activating proper control signals at proper times. There are three groups of control signals.

- **Control signals to be sent to the light NI.** *Accept-test-data* and *valid-test-response* are two signals that are produced by the test block and sent to the light NI. They are one part of handshaking signals between light NI and test block in order to send and receive data. When light NI has test data ready to send to the test block , it activate the *valid-test-data* signal (that is the input of the test block). Whenever the test block receives the test data it activates the *accept-test-data* in order to indicate that it has received the test data.

  On the other side, for sending the test response to the light NI, whenever the test response is ready to be sent to the light NI the test block activates the *valid-test-response* signal. After the light NI receives the test response it

sends an acknowledge signal as *accept-test-response* to the test block. After the test block receives the *active accept-test-response* signal, it deactivates the *valid-test-response* signal.

- **Control signals for shift registers.** There are two shift registers in the wrapper design. One at one of the inputs of the router-under-test and one at one of the outputs of the router-under-test. There is a single register-module that can function as the input shift register or as the output shift register. Only by receiving proper control signals at each time the shift register can have the correct functionality, either as **input shift register** or **output shift register.**

  If the register-module is an input shift register, it will act as a parallel to serial converter. It receives parallel data and sends it out bit by bit. If the register-module is an output shift register, it will act as a serial to parallel converter. It receives data bit by bit and it send the data out in parallel after number of clock cycles.

  The test block in order to define the functionality of each instance of the register-module produces 6 control signals as input to this register-module.

  - **Serial-out.** When serial-out signal is active, the content stored in the register will be sent out bit by bit.
  - **Serial-in.** When scan-in signal is active, it receives one bit as input at a each clock cycle.
  - **Parallel-load-in-input-reg.** By activating this signal parallel data will be loaded into the register as input register.
  - **Parallel-load-out-input-reg.** Activating this signal leads to load out parallel data from the register as input register.
  - **Parallel-load-in-output-reg.** The same as parallel-load-in-input-reg, this signal also enables loading parallel data to the register, but as output register.
  - **Parallel-load-out-output-reg.** The same as parallel-load-out-input-reg, this signal also enables loading out parallel data from the register, but this time as output register.

- **Control signals sent to the router-under-test.** *Scan-enable* and *hold* are the two signals that are sent to the router-under-test. When *scan-enable* signal is active, the internal flip-flops of the router convert to scan flip flops and form the scan chain in the router.

  Hold signal is an active-low signal, which means when it is active it is "0". As depicted in Figure 5.6, the *hold* signal will be *Anded* by the *clock* signal and produces an internal clock signal in the wrapper named as *clock-w*. *Clock-w* is used as clock signal for the router-under-test, the input-register and the output-register. Therefore, whenever the *hold* signal becomes active ("0"), the *clock-w* signal becomes "0". The router-under-test and the registers at input and output of the router-under-test are all working with rising-edge of their input clock signal. Therefore, when their input clock signal (clock-w)

becomes "0", their functionality will be halted until they receive a rising-edge again. In other words, the registers and the internal flip-flops in the router-under-test will hold their current content until the *hold* signal become deactivated (becomes "1") again.

For activating proper control signals at proper times, the test block should know at each time that at which step of testing procedure are we. The test procedure is composed of 6 sequential steps. Meaning that they should keep their order. In what follows the sequence of steps and the control signals that should be activated at each step are explained.

(a) Inserting sequential test stimuli to the scan chains of the router. First the test message should be received from light NI and *accept-test-data* becomes *active*. Afterward, control signals that should be activated are :
- *Parallel-load-in-input-reg* signal to enable loading test stimuli in the input register.
- *Serial-out* signal in order to activate parallel-to-serial mode of the register placed at input port of the router-under-test.
- *Scan-enable* signal in order to enable test stimuli insertion to the scan chains of the router-under-test.

(b) scanning out the sequential test response from scan chains of the router. Control signals that should be activated are:
- *Scan-enable* signal should be active in order to be able to scan out the contents of the scan chains of the router-under-test.
- *Serial-in* signal has to be active to enable serial-to-parallel mode of the register that is placed at the output of the router-under-test.
- *Parallel-load-out-input-reg* signal enables loading the test response in pieces of **word**.
- *Valid-test-response* signal, informs the light NI that there is a data (test response) to be sent to the light NI in order to get packetized.

(c) inserting sequential data for initialization to scan chains of the router. In this step like the first step after receiving the data in for of message from the light NI, the *accept-test-data* should become active. Other required active control signals are:
- *Parallel-load-in-input-reg* signal to load the test data to the input register.
- *Serial-out* signal to activate parallel-to-serial mode at the input register.
- *Scan-enable* signal to enable scanning in initialization data to the scan registers of the router-under-test.

(d) Sending parallel data to one of the functional inputs of the router. At this step test stimuli should be sent to the functional port of the router in order to test the combinational logic part. Therefore, like step 1, first the test stimuli should be received from the light NI and the *accept-test-data* signal should become *active* to acknowledge the light NI that the test stimuli has been received. Other control signals that should be activated at this step are:

- *Parallel-load-in-input-reg* signal to enable loading of the parallel test stimuli to the input register.
- *Parallel-load-out-input-reg* signal to enable loading out of parallel data from the register that is at input port of the router-under-test to the functional input of the router.

(e) Receiving parallel data from one of the functional outputs of the router. Similar to the previous step the required active control signals in this step are:

- *Parallel-load-in-output-reg* signal enables loading parallel test response (related to testing the combinational part) from functional output port of the router-under-test to the register at the output port.
- *Parallel-load-out-output-reg* signal should be activated to enable loading out the test response from the output register to the light NI.
- *Valid-test-response* signal also should be activated to inform the light NI that a test data is ready to be sent to the light NI to get packetized.

(f) Scanning out the sequential test stimuli from the scan chains of the router. Control signals that should be activated at last step are:

- *Scan-enable* signal should be active at this step in order to be able to scan out the content of the scan chains of the router-under-test.
- *Serial-in* signal enables serial-to-parallel mode of the output register.
- *Parallel-load-out-input-reg* signal enables to load out the parallel test response (that is the contents of the internal flip-flops after test stimuli application) from the output register to the light NI.
- *Valid-test-response* signal should become active at this step to notify the light NI that test response is ready to be sent to light NI for packetization.

Moreover, If testing the router-under-test is in one of the steps of 1,2,3 or 6, there is a need to take care of continuity of the test data either if it is test stimuli or test response. This is the test block responsibility to keep the sequential test stimuli and test response like streaming data. This is needed only in phases that we are sending or receiving data to the scan chains. For this reason the test block has a control signal names as *'hold'*.

As mentioned before the test stimuli should be transported in packets. The test packets are transported in NoC using BE connections which the arrival time of the packets are not known. Therefore, there may exist interrupts in between different segments of the test stimuli that are transported by different flits of the packet [25].

When receiving a test stimuli to scan chains or sending a test response from the scan chains starts, if in the middle there is an interrupt in receiving of test stimuli, the *'hold'* signal should become activate . The *'hold'* stays active until the next part of test stimuli arrives (or next part of test response can be transferred).

[25] BE packets in AEthereal are using wormhole routing

5.4.3 Test block implementation

As it is depicted in Figure 5.12, the test block is responsible for activating control signals and transferring the test-data-in to test-data-to and test-response-from to test-response-out at proper time. The proper time for these thing to happen is decided by the control part of the test block.

5.4.3.1 Control counters and signals

There are 3 counters in the design for controlling the amount of inserted test stimuli or the amount of test response that has been scanned out. The controlling counters are:

- **Bit-counter-in.** It counts the number of bits in a word to understand when the word is completely sent to the scan chain. The maximum possible value of this counter is the number of bits that exist in a word of test message (32).
- **Bit-counter-out.** It counts the number of bits in a word to understand when the word is completely received from the scan chain.
- **Test-pattern-word-counter.** This counter counts the number of words that exist in the current test pattern that is either sent to the scan chain or to the functional input of the router-under-test. The number of words that a test pattern has, should be defined by the TPG and given to the wrapper by using *Generics*. When this counter reaches its maximum number that is the real size of a test pattern in words, it means that the scanning in procedure has been finished. The maximum value of this counter (number-of-test-pattern-words) is given by Generics to the design.
- **Scan-chain-word-counter.** This counter counts the number of words in the scan chain that has received test stimuli or, when scanning out, the number of words in scan chain that the data has been scanned out. When this counter reaches to its maximum number it means that the scan chain in the router-under-test is full and is ready to scan out the data words. The maximum value of this counter (number-of-scan-chain-words) is given by Generics to the design.

There are also two internal control signals in the test block that defines at each time if the input and output registers are empty or not.

- **Reg-in-empty.** This signal shows if the input register is empty or full.
- **Reg-out-empty.** This signal defines whether the output register is full or empty.

If one of the signals above is set to "1", it means that the corresponding register is empty. When the register is full the corresponding signal is set to "0". The input register and the output register, both have one word capacity. The way that they become full or empty is explained in more detail in section (shift register). These

two signals are registered and at rising edge of each clock cycle they can receive a new value. In the beginning with the reset signal, at the rising edge of clock, they will be initialized to "1" which means initially the registers are empty.

5.4.3.2 Control block

The control block is a Mealy state machine. The state machine, based on the input signals from the light NI, contents of the internal control counters, internal control signals and the current state, decides which control signals have to be activated and what is the next state of the state machine.

With the *reset* signal the state machine enters to the state 1. It will remain at state 1 until the test block receives the first active *valid-test-data* input signal from the light NI. The first *valid-test-data* signal defines the beginning of the test procedure. After receiving the first *valid-test-data* signal the state machine will turn on. At each state of the Mealy machine it is known that at which of the 6 steps of testing we are. Therefore the corresponding control signals to that step of testing will be produced in the corresponding sate of the Mealy machine.

The block digram of state transition in the proposed Mealy machine is depicted in Figure 5.13.

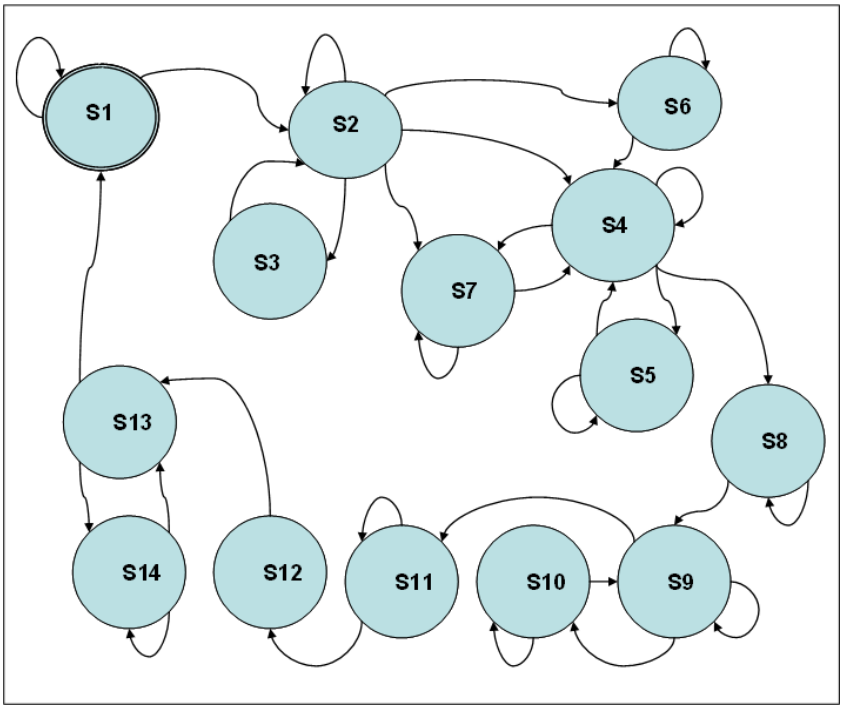


Figure 5.13: The block diagram of the state machine

As depicted in Figure 5.13. The state machine is composed of 14 states. The first 7 states of the state machine are related to the first phase of router test which

is testing internal flip-flops of the router-under-test. This phase of testing consist of scanning-in the test stimuli to the scan chain and scanning-out the contents of scan-chain.

The states 8 to 10 are related to inserting an initialization vector to the scan chain of the router. States 11 is the state to receive and load the parallel test stimuli to the functional input port of the router-under-test. State 12 is the state that the parallel test response is received from one of the functional output ports of the router and sent to the output register. State 13 and 14 are the states that the contents of scan chain in the router-under-test after applying the test stimuli, will be scanned-out.

In what follows, the necessary control signals that become activate in each state transition and the state transitions in each state are explained.

(a) **State 1.** This is the starting state of the machine. At this state, test block is waiting to receive the first *valid-test-data* signal from the light NI. This means that, the state machine will remain at this state until it receives an active *valid-test-data* signal. In case a *valid-test-data* signal is received, the state machine starts working and will go to next state, State 2.

When a *valid-test-data* signal is not received yet, the state machine is not working yet, because this means there is no test stimuli received by the wrapper yet. Therefore, there is no control signal that should be activated at this point.

At the time that *valid-test-data* is received it means that the first test stimuli word is ready to be received from the light NI. Since, it is the starting state of the state machine and the first word of test stimuli is being received, we can be sure that the input register is empty. Therefore the ready test stimuli word from the light NI can be loaded to the input register. But the next clock cycle the input register should not be empty. The control signals that should change value are:

- **Parallel-load-in-input-reg-next.** This signals should become active to be able to load the ready test data to the input register.
- **Accept-test-data-next.** This signal becomes active to acknowledge the light NI that it has received the data.
- **Reg-in-empty-next.** This signal becomes "0", to show that the next clock cycle the input register is not empty anymore.
- **Test-pattern-word-counter.** This counter should be incremented by one. Because at this point one word of test pattern has been received.

(b) **State 2.** This state is the scanning-in state. In other words, it is for transferring test stimuli to the scan chains. Whenever a new test stimuli word that should be sent to scan chains of the router is received the next state is state 2. Therefore, the state machine is at different parts of the scanning-in process is in this state. In other words, when state machine is in state 2, we can be at the beginning, in the middle or at the end of scan-in process.

In the process of scanning-in one word of test stimuli, one bit at a time will be scanned-in to the scan chain. For this reason the control signals that should be activated are:

- **Serial-out-next.** This signal becomes "1" to enable parallel to serial mode of the input register.
- **Scan-enable-next.** This signal becomes "1" to enable scan mode of internal flip-flops of the router and form the scan chain.
- **Count-scan-bit-next.** This counter has to increment by one. Because one bit of the test stimuli word that is stored in the input register is being scanned-in to scan chain.

the next state will be again state 2, until count-scan-bit = 32 (number of bits in a test stimuli word). At this point three different values should be checked to decide what are the active control signals and state in next clock cycle. The values are :

- **Test-pattern-word-counter.**
- **Scan-chain-word-counter.**
- **Valid-test-data signal.**

The counter test-pattern-word-counter, can define if the scan-in process has reached to end or if it is still in the beginning or in the middle. If the content of the counter is not equal to the maximum number of test pattern words, then it means that the scan-in process is still in the beginning or in the middle. It means that there is still test stimuli words to expect.

When the test block is in the beginning or in the middle of scan-in process, it can still expect new test stimuli words to be received by the light NI. Therefore, still in this state the *valid-test-data* should be checked. If *valid-test-data* is "1", it means there is a new test stimuli word ready by the light NI to be sent to the test block. In this case the next state is again state 2, because a new test stimuli word is ready to be accepted.

If *valid-test-data* signal is "0", it means the next word of test stimuli is is not ready (the test packet containing it has not been arrived yet). Therefore the scan chain of the router and the shift register at the input should be halted, so they keep their current value. Therefore the *hold* signal should become active ("0"). Moreover *hold* signal should stay active until an active *valid-test-data* signal is received. Therefore the next state is state 3 which is the wait state for *valid-test-data*.

When we are still in the beginning or middle of the scan-in process, if scan-chain-word-counter has its maximum value, it means that the scan chain has been already fulled. But the test pattern has not been finished yet. Therefore, the next state test block should be able to scan-out test response (in order to make space for the new bits of test stimuli)and at the same time scan-in the new bits of test stimuli. This state will be state 7.

If the content of the test-pattern-word-counter reaches to the maximum number of words that the current test pattern has, test block understands that it

the the end of scan-in process and next state is either state 4 or state 6. In this case, there is no need to check *valid-test-data* signal anymore. Because there is no other test stimuli words to be received.

The condition that decide whether the next state is state 4 or state 6 is the content of scan-chain-word-counter. If the scan-chain-word-counter is equal to the maximum number of words in the scan chain, it means that the scan chain is full. Therefore, the next clock cycle a valid test response will be shifted out from the scan chain. In this case the next state is state 4, which is the scanning-out state.

If the scan-chain-word-counter has not reached its maximum value, it means that the scanned-in data has not filled the scan chain. Therefore, if we try to scan-out data the next clock cycle, the test response bit that is scanned-out is not valid. This means that first, the contents of the scan chain should be shifted until they reach to the end of scan chain and then they can be scanned-out as valid test response. State 6, is the state that just shifts the contents of the scan chain toward the scan-out pin. Therefore, in this case the next state is state 6.

When test-pattern-word-counter = maximum-number: 1. When scan-chain-word-counter = maximum-number 2. When scan-chain-word-counter ¡¿ maximum-number

(c) **State 3.** This state is the state that waits for the next test stimuli word during scanning-in process. The only condition that will be checked is whether *valid-test-data* is "1" or not. If it is "1" the next state of the state machine will be state 2 to scan-in the new test word that is ready. But if the *valid-test-data* is not active, the next state will be state 3 again.

At the time of waiting for *valid-test-data*, the *hold-next* control signal, should be active ("0"), *shift-input-reg-next* and *scan-enable-next* should be de-active "0". Hence the scanning-in process from input shift register to the scan chain of the router-under-test will be halted. By activating *hold*, the scan chain and input register can hold their contents unchanged until the next word of test stimuli arrives.

(d) **State 4.** In this state the contents of the scan chain will be scanned-out and shifted to the output shift register. But if the number-of-test-pattern-words is greater than the number-of-scan-chain-words, it means that although the scan chain is full but there are more test stimuli words waiting to be scanned-in to the scan chain. Therefore, in this case the scan chain should have the ability of both scan-in and scan-out at the same time.

When the number-of-test-pattern-words is greater than the number-of-scan-chain-words, two different scenarios may happen. Either the test-pattern-word-counter has reached to its maximum possible value or it has not. If it has reached to the maximum value, it means that the test stimuli has finished and the last test stimuli word is in the input shift register while the scan chain in full. But if the test-pattern-word-counter has not reached to its maximum possible value, it means in addition to the test stimuli bits that are already

in scan chain and in input shift register, test block should expect more to come. Therefore, at this situation, test block should still check for an active *valid-test-data* signal.

Moreover in all scenarios explained above, it should always be checked if the output register has enough space for receiving the next test response bit or not. In other words if the *reg-out-empty* is "1" or not.

The two main conditions that should be checked first are checking the value of test-pattern-word-counter and checking if the output register has enough space or not. In different combination of these 2 conditions we may need to check other conditions such as if (the number-of-test-pattern-words is greater than the number-of-scan-chain-words). When the test-pattern-word-counter has not been reached to its maximum value, we do not need to check the condition if the number-of-test-pattern-words is greater than the number-of-scan-chain-words. Because the scan chain is already full but test pattern has not been finished. Therefore, it is already obvious that the number-of-test-pattern-words is greater than the number-of-scan-chain-words.

The states that the test block can enter after this state are:

- State8: When, the test block is sure that the scanning-out process of first phase of test (testing the internal flip-flops) is finished, the next state will be S8. It can realize the end of scan-out process by checking the value of test-pattern-word-count-out. If this counter has the value equal to number-of-test-pattern-words, it means that the whole phase of testing internal flip-flops has been finished.
- State 5: The state machine will enter to state 5 (wait state for output register becoming empty), in two different cases. First, is when the output register is full (reg-out-empty = "0").
  When the last bit of the current test response word has been scanned-out and shifted in to the output shift register. At this point we are sure that the output register is full. At this point if the test-pattern-word-count-out counter has not reached to its maximum value, it means there are more test response words to be scanned-out. Therefore the second situation that the next state from state 4 is to enter state 5 is here. Because, at this situation the test block is sure that the output register is full, but there are more test response words that should be shifted in to the output register. Therefore, test block should enter to state of wait for emptying output register.
- State4: The scanning-out process from the scan chain and shifting-in to the output shift register is done in state 4, bit by bit. At each clock cycle only one bit of a test response word will be inserted to the output shift register. Therefore after 32 (the number of bits in a word) clock cycles the output register will be full and its contents should be loaded out in parallel to the light NI. This means that for 32 clock cycles the state machine should be in state 4 that is the scanning-out state. To check whether a test word has been completely scanned-out or not we can use

the scan-bit-counter. If it has not reached 32 (the number of bits in a word), the next state is again state 4. But if it has reached the value 32, it means that the complete word has been scanned-out and the next state is either S5, S7 or S8 (depending on whether the test pattern words are finished or not and if the test response words are finished or not).

- State7: When the test pattern is not finished yet, the test block expects to receive more test stimuli words. Therefore when a whole test response word has scanned-out from scan chain to the output register, the test block knows that as the output register has been fulled the input register has been emptied. Because, as the test response bits were being scanned-out from the scan chain the test stimuli bits (from the input register) were being scanned-in to the scan chain. Thus we should wait until a new test stimuli word is received from the light NI. Otherwise, if we continue the scanning process, since there is no valid test stimuli word in the input register to be scanned-in to the scan chain, the test will be corrupted. Thus the next state should be the state that waits for a new test stimuli word (*valid-test-data* = "1"), in the middle of scanning-out process. This state is state 7.

(e) **State 5.** State 5 is the wait state for the output register to become empty again, in order to continue the scan-out process. Therefore, in this state the test block is waiting to receive an active accept-test-response. The active accept-test-response signal is a sign that the light NI has accepted the test response word and the output register has become empty. Therefore, until accept-test-response is "0", it means that output register is not empty yet and the next state has to be state 5 again. At this situation the *hold* signal should be activated, so the scan chain and input/output registers preserve their current contents. Moreover the control signals *serial-out* of input register and *serial-in* of output register should be deactivated.

When an active ("1") accept-test-response signal has been received, the next state will be state 4, in order to continue the scan-out process. At this point the *reg-out-empty* control signal should become "1", as a sign that the output register is empty again. The control signals *hold*, *serial-in* and *serial-out* will keep their current values that is halting the scan process. Because only after going to state 4, the scan process should be turned on again.

(f) **State 6.** This state is the state that the contents of the scan chain will be shifted toward the output without receiving any new test data from scan-in pin. This is needed whenever the number-of-scan-chain-words is bigger than the number-of-test-pattern-words. In this situation, the test pattern will finish but the scan chain is not fulled yet. Therefore the number of shifts that are needed until the first bit of test response reaches to the output pin is:

$$number of shifts = (number of scan chain words - number of test pattern words) \times 32$$

This means that for number-of-shift times the next state should be state 6 again. The number of shifts that has been done at each time can be calculated

using the same bit-counter that was used for scanning-in bits. During these number-of-shift times the control signals that should be active or inactive are:

- **Scan-enable-next.** This signal should be active. So each of the internal flip-flops of the router that form the scan chain, shift their contents to the next flip-flop in the scan chain.
- **Serial-out.** This signal should be inactive. Because the test stimuli has been finished and there is no more bits that should be inserted to the scan-in pin. Therefore, by inactivating this signal, the input shift register does not send serial data to the scan-in pin anymore.
- **Hold.** This signal at the time that the scan chain is being shifted should be inactive ("1").

After the counter value reaches to the number-of-shifts, the next state should be state 4 (scanning-out state). At this time the active and inactive control signals are:

- **Scan-enable-next.** Since the next state is also scanning-out the contents of the scan chain. Therefore, this signal needs to be active again.
- **Shift-input-reg-next.** This signal should be inactive again, since there is no new bits of test stimuli to be inserted to scan-in pin.
- **Hold.** Since at this point the test responses should be scanned-out to the output register, first the test block should make sure that the output register has enough space. Therefore, the *hold* signal should be active, so the flip-flops of the scan chain keep their current value.

(g) **State 7.** In this state the test block is waiting to receive an active *valid-test-data*, while it is in the middle of scanning-out process. The test block enters this state only if the test pattern is bigger than the scan chain and at the scanning-out process, new test stimuli bits should also be scanned-in.

Until the *valid-test-data = "0"*The next state will be again state 7 and the scan process is halted which means the active and in-active control signals are:

- Hold-next is active.
- Serial-in-next is in-active.
- Scan-enable-next is in-active.
- Serial-out-next is in-active.

When an active *valid-test-data* is received, the new test stimuli word will be received in the input register and the next state will be state 4, to continue the scan process. The control signals to receive the new test stimuli word in input register are :

- Parallel-load-in-input-reg-next is active.
- Accept-test-data-next is active.
- Test-pattern-word-counter is incremented by one.
- Reg-in-empty-next is set to '0', to show that the input register is not empty anymore.

At this point the test response words will be transferred to the light NI, get packetized and sent to the TRA. If the internal flip-flops are faulty, the TRA will recognize it at this phase and the router will be marked as faulty. In case the router-under-test fails the test at this phase, there is no need to test the combinational logic part of the router. Therefore the test process of the router-under-test will be terminated. The test termination is done by adding the *id* of the router-under-test to the list of defected routers and by ending the test mode of the wrapper.

If the internal flip-flops are fault free the router-under-test passes this phase of test and it will continue to next phase of test process that is testing combinational logic part of the router-under-test. For this reason, it receives the next test data that is an initialization vector. The initialization test vector should be inserted to the scan chain of the router-under-test.

Therefore, after state 7, if the wrapper around the router-under-test receives a un-active test mode signal, it means that the internal flip-flops were faulty. But if after state 7 the wrapper stays at test mode and receive a new test packet, it means that the router-under-test has passed the first phase of test. The test packet that is received contains the initialization vector for internal flip-flops.

In what follows the rest of the states of the Mealy machine will be explained.

(a) **State 8.**From the above mentioned, The test block knows that if after state 7 it receives a new test stimuli word, the test stimuli word is for internal flip-flop initialization and it should be scanned-in to the scan chain. Therefore the state 8 of the state machine in test block will be again the waiting state to receive the first word of initialization vector. State 8 is exactly the same as state 1. Both of them are the waiting states to receive the first word of the test data that should be scanned-in to the scan chain later. The only difference between these two states is the next state that the state machine should enter after them. The next state of state 8, is state 9.

(b) **State 9.**In state 9 the first word of initialization vector that is loaded to the input register should be serialized and scanned-in to the scan chain of the router-under-test. Therefore state 9 is almost equal to state 2. In both of these states the test data word that is in the input register should be scanned-in to the scan chain, while there is a possibility to receive a new test data word. One major difference between state 9 and state 2 is that, in state 9 the test data that is being scanned-in to scan chain in the test vector for initialization the scan chain. Therefore the number of bits of test vector is exactly the same as number of bits in the scan chain. However, this was not the case in state 2. In state 2 the test stimuli pattern can have smaller, equal or bigger size than the scan chain. Therefore, in state 2 there was a need to differentiate between the three cases that test stimuli pattern was smaller,equal or bigger than the scan chain. But in state 9 there is no need for such differentiation. Another difference between these two states is the next state that they will enter. State 9 after scanning-in the complete initialization vector to the scan chain should enter to a state that waits to receive the first word of combinational-test-

stimuli. This state is state 11. State 9 will enter state 10, when there is a need to wait for receiving the next test vector word for initializing the scan chain.

(c) **State 10.** The wait state for receiving the next test vector word is state 10 which is equal to state 3. Only their next states are different. State 10 after receiving the new test vector word will return to state 9 for scanning-in the new test vector bits.

(d) **State 11.** In state 11 the test block is waiting to receive the only test stimuli word that should be applied to one of the functional inputs of the router-under-test. The test stimuli that should be applied to the functional input is only one word. It has the size equal to the size of functional ports of the router. Therefore in state 11 the state machine is waiting for *valid-test-data* signal to receive the first and only word of expected test stimuli. Until *valid-test-data* is inactive ("0"), the next state is again state 11. When an active *valid-test-data* is received the next state will be state 12.

(e) **State 12.** In state 12, the test response of combinational logic test will be received from one of the functional output ports of the router-under-test. In this state the test block is sure that the output register is completely empty. This is due to the fact that the previous data that was stored in output register was the test response of testing internal flip-flops. The test block is sure that this data has been loaded out from the output register. Because this data had to be checked first by the TRA, in order to test block to receive the test data related to testing the combinational logic part. Therefore in this state the test block should just activate the control signals for parallel loading the test response to the output register and then parallel loading out the contents of the output register to the light NI. Therefore the control signals that should be activated are:

- parallel-load-in-output-reg
- parallel-load-out-output-reg
- valid-test-response

The output register will not be signed as full, because at the same clock cycle that the data is being transferred to the output register, it is also being loaded out from the output register. The next state after this state will be state 13 for scanning-out the content of the scan chain again.

(f) **State 13.** State 13 is almost similar to state 4. Both of them are states that the contents of the scan chain will be scanned-out and inserted to the output register. In state 13 the contents of the scan chain after the test stimuli is applied to the combinational logic part, will be scanned-out. Therefore, in this state, the test block is sure that while it is scanning out the contents of the scan chain, there is no new data that has to be scanned-in to the scan chain. When the scanning-out process in this state finishes the state machine will return to the first state (state 1) again. If in the middle of scanning-out, the output register becomes full, then the state machine should enter to a

state that waits for output register to become empty again. This state is state 14.

(g) **State 14.** State 14 is the state that the test block should wait for an active *accept-test-response* signal from the light NI. Receiving this active signal means that the light NI has accepted the test response word that was ready in the output register and the output register has become empty. In this case the next state will be state 13.

5.4.4 Initialization of the test block

With receiving an active *reset* signal, the test block will initialize its control signals and its counters. In the initialization the control signals will be initialized to their inactive value and the counters will be initialized all to 0. The state register will be initialized to state 1.

5.5 The shift registers

The shift register module can act as input register (at the input port of router-under-test) or output register (at the output port of router-under-test). The capacity of the registers is one word (32 bits). This register can have 3 different operational mode:

- **Parallel-to-serial converter.** In this mode, it receives parallel data but the output is sent out bit by bit. For having this operational mode the control signals *parallel-load-in-input-reg* and *serial-out* should be active. This mode is useful only in the input register in the wrapper.

  After receiving first word of test stimuli the input register becomes full (reg-in-empty = "0"). When we are in scanning-in steps of the testing, the input register only becomes empty after all the stimuli bits are shifted out of the shift register. Considering the size of the input register is 32 bits, this will take 32 clock cycles. Because one bit of the test stimuli that is in the input register will be shifted out at each clock cycle. Count-bit-in keeps track of number of bits of test stimuli that has been shifted out from the input register and scanned-in to the scan chain of the router-under-test. Therefore whenever the counter count-bit-in has the value equal to 32, it means that the input register has been emptied again. Therefore whenever count-bit-in become 32, the reg-in-empty can be set to "1" again.

- **Simple register.** This mode can happen both in input register and in the output register. In this mode the parallel data will be loaded in the register and the next clock cycle the parallel data will be loaded out of register. The control signals that should be active in this mode can be from two sets.

  (a) **Set one.***Parallel-load-in-input-reg* and *parallel-load-out-input-reg* should be active together.

(b) **Set two.***Parallel-load-in-output-reg* and *Parallel-load-out-output-reg* should be active at the same time.

- **Serial-to-parallel converter.** This mode is used in the output register. In this mode the data is received bit-by-bit, and after the complete word is received, the complete word is loaded out of the register. The signals *serial-in* and *parallel-load-out-output-reg* should be active in this mode.

  The output register the same as input register is initially empty. But in the output register, at the scanning-out steps, the test response enters the output register serially (one bit at each cycle). Therefore, Considering a 32 bits output register, it will have enough space to receive serial test response for 32 continuous bits. This means that the output register after receiving 32 bits of serial test response that takes 32 clock cycles will be full (reg-out-empty = "0"). The counter count-bit-out keeps track of number of test response bits that are scanned-out from the scan chain of the router to the output register. Therefore whenever count-bit-out becomes 32 the output register becomes full. After one parallel load out the whole word of test response will be loaded out from the output register and it will be come empty again (reg-out-empty = "1").

It should be noticed that these two sets are equal. Both of them set registers in the register mode. Just there are two different signal groups for these because the input register and output register need to load in data and load out data at different times. Therefore there should be two signals because there is only one module that serves as input reg and output register.

5.6 Simulation results

The control part of the wrapper, which is the test block is designed completely in VHDL. The design of test block is simulated by using a testbench. The testbench is used to give virtual data and control signals as input to the test block, in order to observe the outputs ans state of the test block at each time. If the test block goes to correct state and generates correct outputs at each time, depending on the input that the testbench provides for it. In order to show the correctness of the designed test block, some of the simulation results are shown in what follows.

In this simulation, the length of the test pattern is considered to be 3 words and the length of the scan chain is considered to be 5 words.

Figure 5.14 depicts the behavior of the test block in the beginning of test. During reset, all signals and counters will be initialized to their initial value and state machine will be in state S0. After reset, the state machine enters to state S1. It stays in state S1, until the first words of test data will be on the test-data-in lines and the signal valid-test-data becomes active for the first time. At this time test block enters state S2. This is depicted by a red circle in the wave form.

In Figure 5.15, depicts the test block behavior while it is still in state S2. It is depicted that test block state for 32 (equal to the length of one word) clock cycles

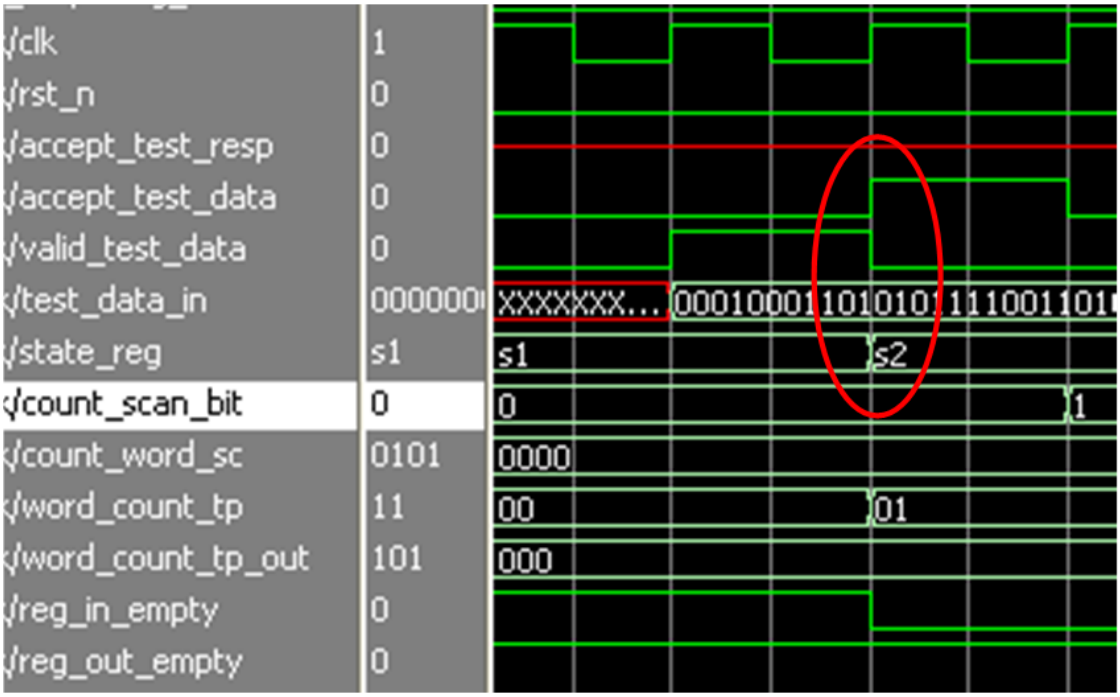


Figure 5.14: Behavior of state machine of test block in test phase 1

in S2 to scan-in the test data word to the scan chain. After 32 clock cycles, since the counter word-count-tp (that counts the number of words of test pattern that has been inserted) is not equal to 3^{26}, it means that there are still some words of test pattern that should be received and scan-in to scan chain. At this point (it is depicted with first red circle), if the valid-test-data signal is not active (which means there is no new test data word ready), the state machine enters the wait state S3. After receiving the first active valid-test-data, the test block enters S2 to scan-in the new word of test pattern to scan chain (this is depicted by second red circle.).

Figure 5.16, shows the time that the test block has scanned-in all the words in of test pattern to scan chain. This can be denoted by looking at word-count-tp counter that counts for words in test pattern. At this point it is equal to 3. However the other counter that counts the number of words in scan chain that has become full (count-word-sc), has not become its maximal value (5) yet. Therefore, the next state will be S6.

In state S6, the state machine shifts the test data in the scan chain until the test data reaches the end of scan chain and can be scanned-out in next clock cycle. For this reason, in this example the state machine stay in S6 and shifts the content of scan chain for 64 clock cycles. This is because we have inserted 3 words to scan chain, while its length is 5 words. Therefore 2 last words of scan chain is empty

[26] the considered length of test pattern in this simulation.

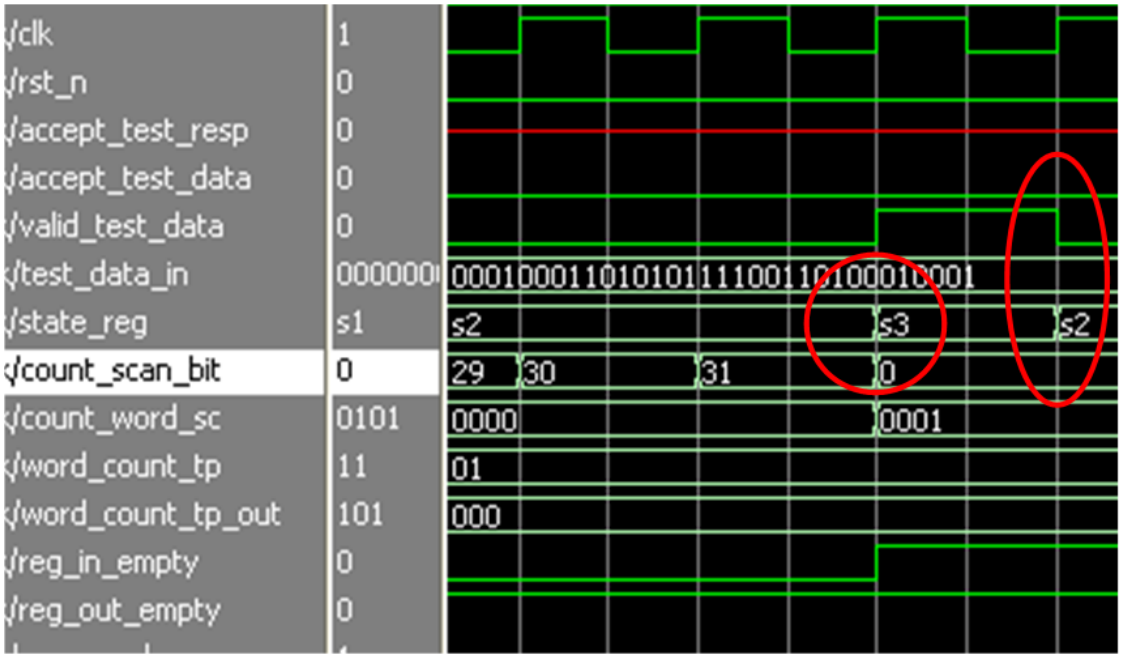


Figure 5.15: Behavior of state machine of test block in test phase 1

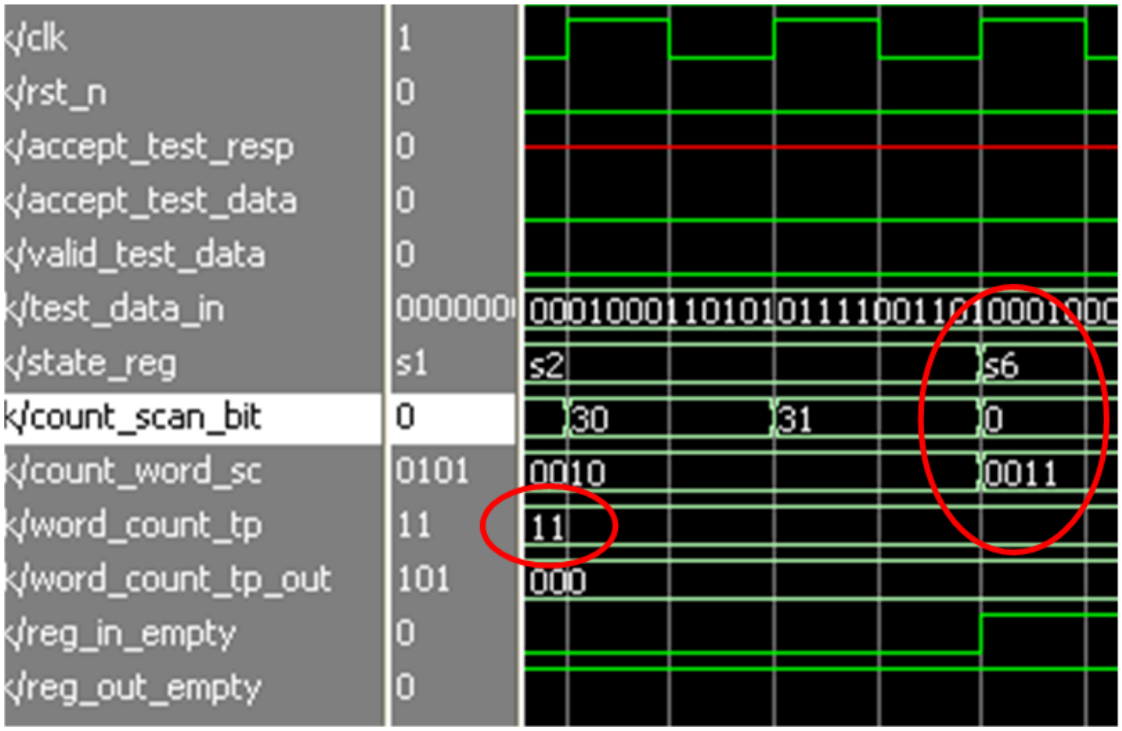


Figure 5.16: Behavior of state machine of test block in test phase 1

that is equal to 64 bits. When the count-scan-bits counter count to 64 and the state machine is still in S6, the contents of the scan chain are at the end of scan chain and are ready to be scanned-out in next clock cycle. Therefore at this point the next state will be S4. This situation is shown in Figure 5.17.

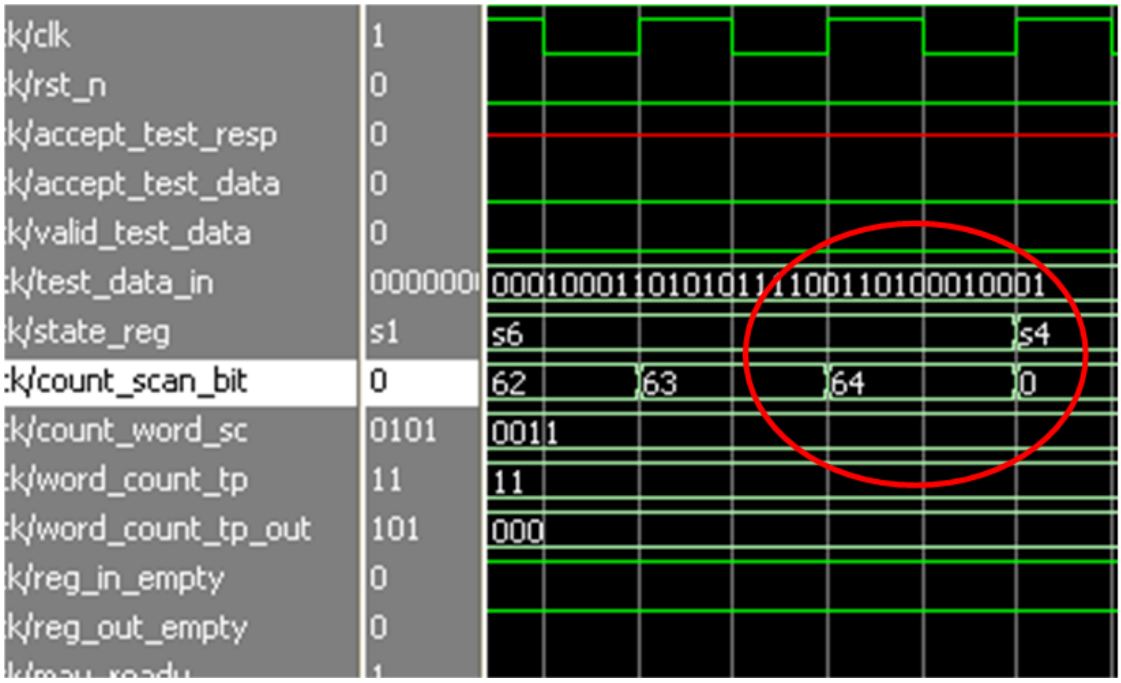


Figure 5.17: Behavior of state machine of test block in test phase 1

State S4 is the state for scanning-out the content of the scan chain to the output register. Figure 5.18, depicts the time that the test block has scanned-out the first word of scan chain to output register. At this point, the test block should wait for an accept-test-response input from the the light NI. This is because the test block should be sure that the first word of test response is received by the light NI and now there is enough space in output register to scan-out the second word from scan chain. While the state machine is waiting for the accept-test-response to become active, it is in state S5 (this is shown by first red circle). In state S5, with receiving an active accept-test-response, in next clock cycle the test block will enter again to S4 to scan out the next word of test response from scan chain.

Figure 5.19 depicts the time that the test block has scanned out all the test response words from the scan chain. This can be denoted by looking at word-count-tp-out that is equal to 3. At this point the next state will be state S8. State S8, is the wait state to receive an active valid-test-data signal for the test data that should be inserted to scan chain for initializing the router in second phase of test. The third red circle shows valid-test-data becomes active and the state machine will enter to S9 to scan-in the initialization vector to scan chain.

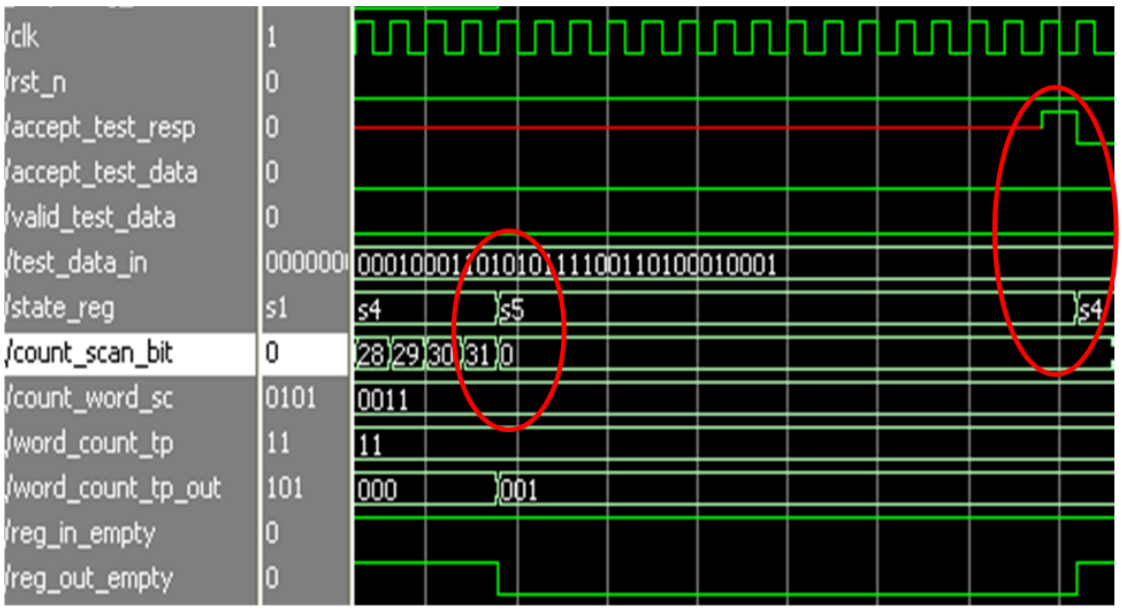


Figure 5.18: Behavior of state machine of test block in test phase 1

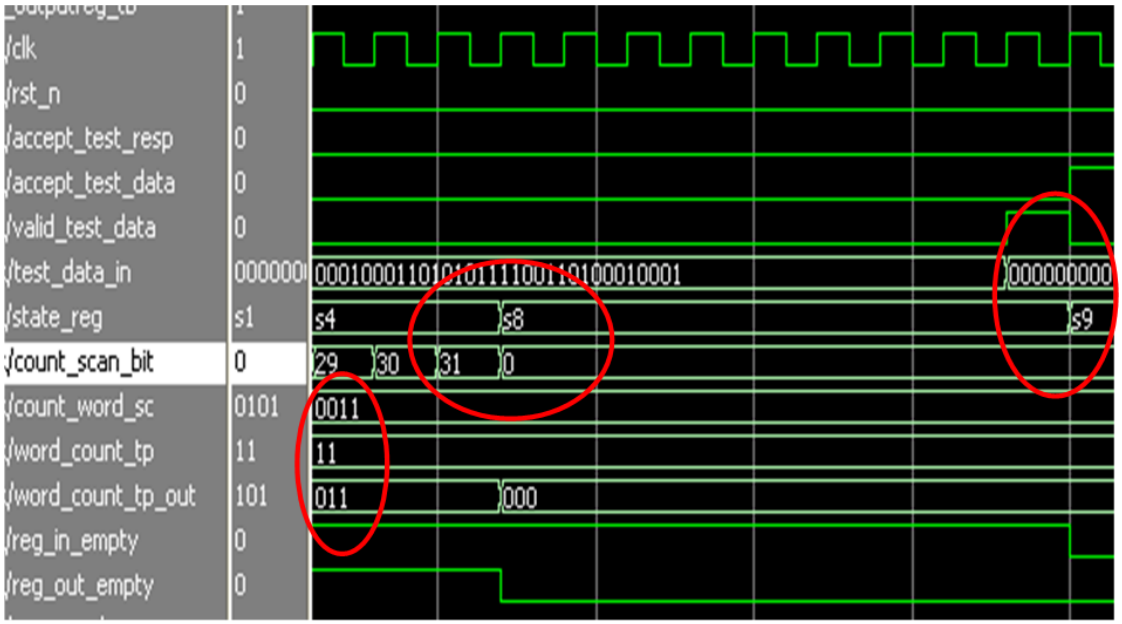


Figure 5.19: Behavior of state machine of test block in test phase 2

In Figure 5.20, the first red circle, shows the time that all of the words of initialization vector has scanned-in to the scan chain. This can be noticed by looking at count-word-sc counter that has the value 5 at the end, therefore the next state

after S9 is S11. At S11, the state machine is waiting to receive an active valid-test-data. This valid-test-data is for receiving the test word for testing combinational logic part of router. At second red circle, the valid-test-data signal has become active and state machine has entered state S12. In S12, the test data word will be applied to one of the functional input ports of the router. After one clock cycle, At S13, the test response of testing combinational part of router is received from one of the functional output ports of the router and sent to the output register. For scanning-out the content of scan chain to the output register, first the test block should know that the previous test response word has been received by the light NI or not. Therefore, at this point, the test block will enter state S14. At S14 it waits until it receives an accept-test-response signal. With receiving an active accept-test-response, test block understands that the output register has become empty. At this point, as depicted with the third red circle, it enters State 13 to scan out the contents of scan chain.

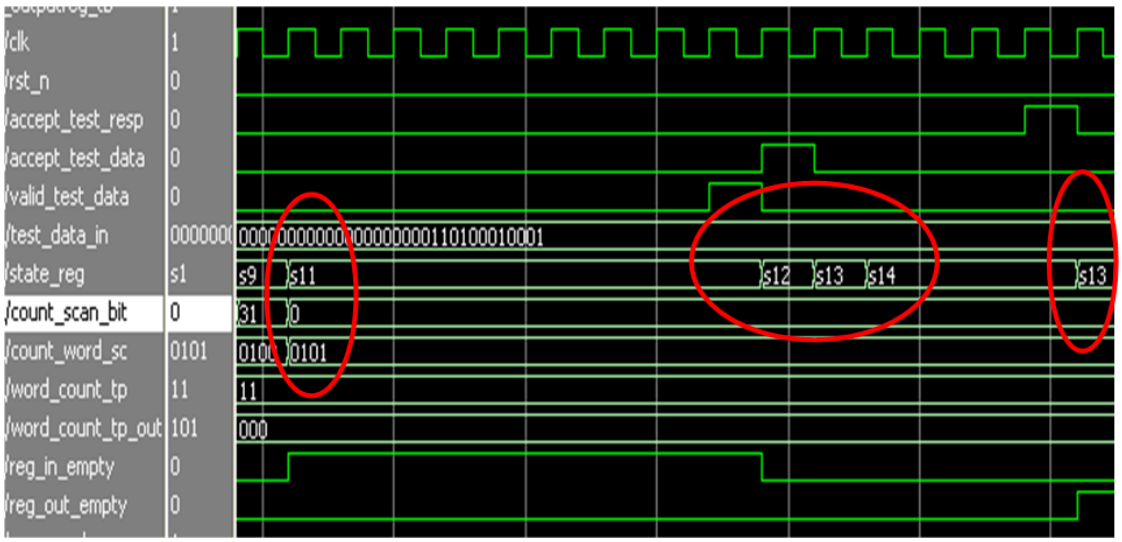


Figure 5.20: Behavior of state machine of test block in test phase 2

Figure 5.21, shows the end of testing process. When all of the test response words are scanned out from the scan chain in state S13, the test process is finished. This can be noticed by looking at the value of count-word-sc counter which is equal to 5 at this point. At this point when state machine knows the test process is finished, if it receives an active *may-ready* signal from the light NI, the output signal *ready* becomes active. In this way the wrapper flags to the scheduler that the test of corresponding router is finished. At the end of test process the state machine will return to state S1.

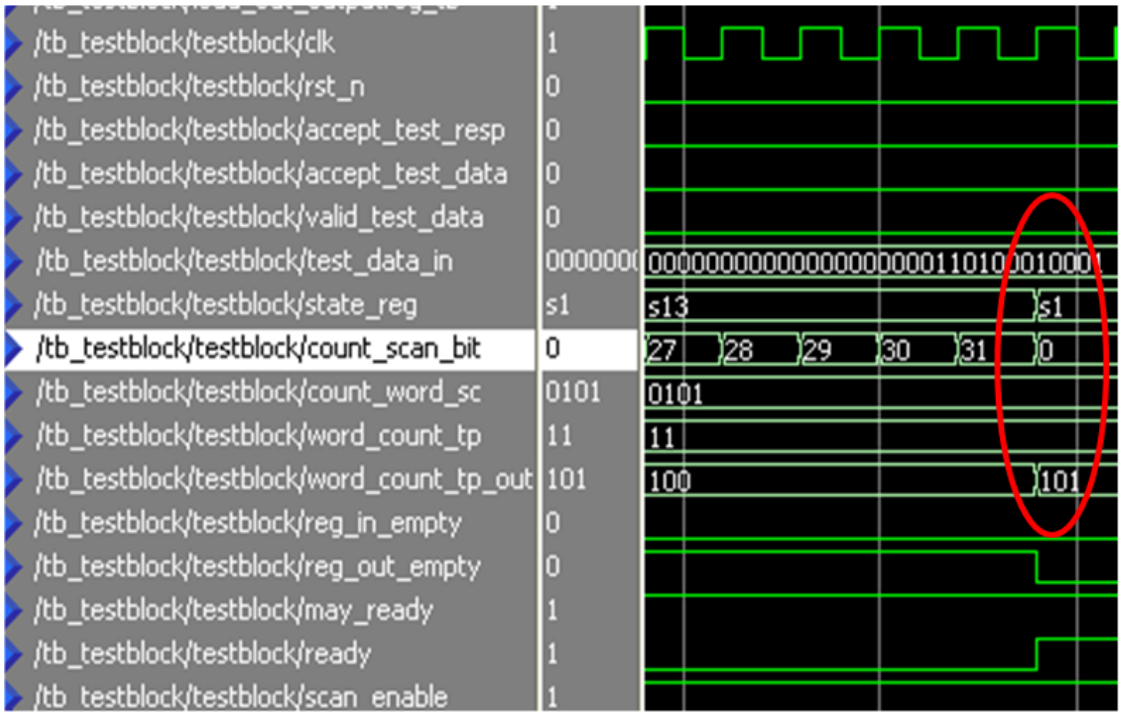


Figure 5.21: Behavior of state machine of test block in test phase 2

6 Conclusion

In this project, an online-structural test method for NoC is explored. In the proposed method, routers of NoC were the targets to be tested. Online-structural test is used to increase the reliability of the NoC. Although the elements of NoC have been tested after manufacturing and before being used in practice, but still after some time that element are they can get defected. Online-structural test help us with detecting and recovering from these types of defect that can happen during the time that NoC is in application.

In the explored method, the inherent hardware redundancy of the NoC is being used. By taking advantage of router redundancy in the NoC, the routers that are free and are not being used in providing connections between on-chip cores, are being tested. This method makes it possible to test the routers of the NoC, while the NoC is running its normal application.

In order to realize online-structural test of router, there is a need to two major factors; A high-level scheduler and a wrapper design. The main contribution of this project was:

(a) Defining the high level design of the scheduler.

(b) Designing the wrapper around the routers.

The key features of the designed wrapper are:

- The wrapper is designed in a way that the NoC can be reused as TAM in the required test architecture and there is no need for extra channels as TAM.
- The wrapper is designed in a way that there is no need to change the router architecture or NI architecture. However the interface of the router and the packet format that is being used should be known to design this wrapper.
- The design is in a way that reusing NoC as TAM does not degrade the performance of the NoC. This is due to the fact that, in this method,test data is being sent by BE connections. Therefore the GS connections are left to be used in providing critical connections in SoC.
- There is no need to program the wrapper. Therefore there is no need for extra dedicated channels to program the wrappers. The only extra hardware over head is two lines that transfer *function/test mode* and *ready* signals to/from the wrappers.

The wrapper is designed for the routers of AEthereal NoC.

In order to achieve a complete practical test system for doing online-structural test on the routers, there are still number of issues that needs more research in the future. These items are:

- What kind of on-chip TPG and TRA can be used in order to be efficient in terms of cost, test time and fault-coverage;
- Defining a scheduling algorithm in high-level scheduler that trades-off between the performance of the NoC and test time;
- How to make the design of the wrapper more modular that does not depend on a NoC characteristics;
- How should the wrapper design change in order to be able to test multiple routers at the same time;
- If it is efficient to use GS connections to transfer the test data in NoC and how should the wrapper design change for this;

Bibliography

[1] T. D Richardson, C.Nicopoulos, D. Park, V. Narayanan, Y.Xie, C. Das and V. Degalahal. "A hybrid SoC interconnect with dynamic TDMA-based transaction-less buses and on-chip networks," In *proceedings of 19th international conference on VLSI Design*, IEEE 2006.

[2] B. Vermeulen, J. Dielissen and K. Goossens. "Bringing Communication Networks on a Chip: Test and Verification Implications", Philips Research Labratoary, vol. 41, no. 9, pp. 74-81, IEEE Sep. 2003.

[3] D. Wingard. "Micronetworks-Based Integration for SOCs", *Design Automation Conf.*, pp. 673-677, 2001.

[4] W. J. Dally and B. Towles. "Route packets, not wires: on-chip interconnection networks", In *Proceedings of DAC 2001.* IEEE, pp. 684-689, June 2001.

[5] L. Benini and G. D. Micheli. "Networks on Chips: A new SoC paradigm", In *IEEE Computer*, vol. 35, no. 1, pp. 70-78, Jan. 2002.

[6] H. Zimmermann. "OSI Reference Model - The OSI Model of architecture for Open Systems Interconnection", In *Communications, IEEE Transactions on [legacy, pre - 1988]*, vol. 28, no. 4, pp. 425- 432, Apr. 1980.

[7] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S.Malik, J. Rabaey, A. Sangiovanni-Vincentelli. "Addressing the System-on-a-Chip Interconnect Woes through Communication-based design", In *Proceeding of DAC 2001*, pp. 667-672, IEEE 2001.

[8] P. Teehan, M. Greenstreet, and G. Lemieux. "A Survey and Taxonomy of GALS Design Styles", In *IEEE Design Test of Computers*, vol. 24, no. 5, pp. 418-428, Sept. 2007.

[9] L. Benini and D. Bertozzi. "Network-on-Chip architectures and design methods" In IEEE *Proc.- Comput.Digit.Tech.*, vol. 152, no. 2, March 2005.

[10] G. De Micheli, P. Pande, A. Ivanov, C. Grecu and R. Saleh. "Design, Synthesis, and Test of Networks on Chips", In *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 404-413, Sept. 2005.

[11] S. Kumar, "On Packet Switched Networks for On-Chip Communications" In *Networks on Chip*, A. Jantsch and H. Tenhunen eds., Place of publication: Springer US, 2003, pp. 85-106.

[12] C. Grecu et al. "Timing Analysis of Network on Chip Architectures for MP-SoC Platforms", *Microelectronics*, vol. 36, no. 9, pp. 833-845, 2004.

[13] F. Karim et al. "An interconnect Architecture for Networking Systems on Chips"', *IEEE Micro*, vol. 22, no. 5, pp. 36-45, Sept. 2002.

[14] L. Benini and D. Bertozzi. "Xpipes: A Network-on-Chip Architecture for Gigascale System-on-Chip", *IEEE Circuits and Systems Magazine*, vol. 4, no. 2, 2004.

[15] A. Radulescu, J. Dielissen, S. Gonzalez Pestana, O.P. Gangwal, E. Rijpkema, P. Wielage and K. Goossens. "An Efficient On-Chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration", vol. 24, no. 1, pp. 4-17, IEEE Jan. 2005.

[16] K. Goossens, J. Dielissen, A. Radulescu. "AEthereal Network on Chip: Concepts, Architectures, and Implementations", vol. 22, no. 5, pp. 414-421, IEEE Sept. 2005.

[17] T. Bjerregaard and J. Spars. "A router architecture for connection-oriented service guarantees in the MANGO clock-less Network-on-Chip", In *proceeding of the Design, Automation and Test in Europe Conference and Exhibition IEEE*, vol. 2, pp. 1226-1231, March 2005.

[18] G. De Micheli and L. Benini. *NETWORKS ON CHIPS: Technology and Tools*, Place of publication: Morgan Kaufmann, 2006.

[19] X. Bai and S. Dey. "High-Level Crosstalk Defect Simulation Methodology for System-on-Chip Interconnects", vol. 23, no. 9, pp. 1355-1361, IEEE Sept. 2004.

[20] M. Cuviello, S. Dey, X. Bai, Y. Zhao. "Fault Modeling and Simulation for Crosstalk in System-on-Chip Interconnects", pp. 297-303, IEEE 1999.

[21] F. Caignet, S. Delmas-Bendhia and E. Sicard. "The Challenge of Signal Integrity in Deep-Submicrometer CMOS Technology", In *Proceedings of the IEEE*, vol. 89, no. 4, pp. 556-573, April 2001.

[22] A.Pereira Frantz, F.L. Kastensmidt, L. Carro, E. Cota. "Evaluation of SEU and Crosstalk Effects in Network-on-Chip Switches", In *Proceedings of the 19th annual symposium on Integrated circuits and systems design*, pp. 202-207, 2006.

[23] R. Baumann. "Soft Errors in Advanced Computer Systems", In *IEEE Design and Test of Computers*, vol. 22, no. 3, pp. 258-266, May 2005.

[24] M. Nicolaidis. "Design for Soft Error Mitigation", In *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 405-418, Sept. 2005.

[25] R. Ubar and J. Raik. "Testing Strategies for Networks on Chip" In *Networks on Chip*, A. Jantsch and H. Tenhunen eds., Place of publication: Springer US, 2003, pp. 131-152.

[26] E. Cota et al. "Power-Aware NoC Reuse on the Testing of Core-Based Systems", In *Proc. Intl' Test Conf. (ITC03), IEEE CS Press*, vol. 1, pp. 612-621, Oct. 2003.

[27] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*, Place of publication: Springer, 2000.

[28] B.G. west. "At-Speed Structural Test", In *ITC International Test Conference*, pp. 795-800, IEEE 1999.

[29] P. Maxwell, I. Hartanto and L. Bentz. "Comparing Functional and Structural Tests", In *ITC International Test Conference*, pp. 400-407, IEEE 2000.

[30] C. Ciordas, K. Goossens, T. Basten, A. Radulescu, A. Boon. "Transaction Monitoring in Networks on Chip: The On-Chip Run-Time Perspective", In *International Symposium on Industrial Embedded Systems*, pp. 1-10, Oct. 2006.

[31] V. Iyengar, K. Chakrabarty, E.J. Mrinissen. "Test wrapper and Test Access Mechanism Co-Optimization for System-on-Chip", In *Journal of electronic testing: theory and applications*, pp. 1023-1032, 2001.

[32] S.K. Goel and E.J. Marinissen. "SoC test architecture design for efficient utilization of test bandwidth", In *ACM Trans. Design Automation of Electronic Systems*, vol. 8, no. 4, pp. 399-429, Oct. 2003.

[33] E. Cota, L. Carro and M. Lubaszewski. "Reusing and On-Chip network for the Test of core-based systems", In *ACM Trans. Design automation of electronic systems*, vol. 9, no. 4, pp. 471-499, Oct. 2004.

[34] A.M. Amory, E. Cota, M. Lubaszewski and F.G. Moraes. "Reducing test time with processor reuse in Network-on-Chip based systems", In *Proc. Integrated circuits and systems design*, pp. 111-116, 2004.

[35] C. Lui, Z. Link and D.K, Pradhan. "Reuse-based test access and integrated test scheduling for network-on-chip", In *Proc. Design, automation and test in europe*, vol. 1, pp. 303-308, March 2006.

[36] A.Larsson, E. Larsson, P. Eles and Z. Peng. "Optimization of a Bus-based Test Data Transportation Mechanism in System-on-Chip". In *Proceedings of the 2005 8th Euromicro conference on Digital System Design*, pp. 403-409, IEEE Sept. 2005.

[37] H.M. Lin, C. Yen, C. Shih and J. Jou. "On Compliance Test of On-Chip Bus for SOC", In *Design Automation Conference. Proceedings of the ASP-DAC 2004. Asia and South Pacific*, pp. 328-333, Jan. 2004.

[38] Frantz, A. P. et al. "Crosstalk- and SEU-Aware Networks on Chips", In *IEEE Desing Test of Computers Magazine*, vol. 24, no. 4, pp. 340-350, 2007.

[39] Grecu, C. et al. "On-Line Fault Detection and Location for NoC Interconnects", In *IEEE International On-Line Testing Symposium*, pp. 145-150, 2006.

[40] Murali, S. et al. "Analysis of Error Recovery Schemes for Networks on Chips", In *IEEE Design Test of Computers*, vol. 22, no. 5, pp. 434- 442, 2005.

[41] Pande, P.P. et al. "Design of Low Power Reliable Networks on Chip through joint Crosstalk Avoidance and Forward Error Correction Coding", In*IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems*, pp. 466-476, 2006.

[42] L.T. Wang, C.E. Stroud, N.A. Touba, *System-on-Chip Test Architectures: Nanometer Design for Testability*, Place of publication: Morgan Kaufmann, 2008.

[43] A. Jutman, R. Ubar and J. Rail. "New Built-In Self-Test scheme for SoC Interconnect", In *Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics*, July 2005.

[44] A. Hassan, V. Agarwal, B. Nadeau-Dostie and J. Rajski. "BIST of PCB interconnects using boundary-scan architecture", In *IEEE Trans. Computer-Aided Design*, vol. 11, no. 10, pp. 1278-1288, Oct. 1992.

[45] J. Koeter, S. Sparks. "Interconnect testing using BIST embedded in IEEE 1149.1 designs", In *Proc. ASIC Conf.*, 1991.

[46] Chan, J. C. "An Improved Technique for circuit board interconnect test", In *IEEE Trans. on Inst and Meas.*, vol. 41, no. 5, pp. 692-698, Oct. 1992.

[47] C.P. Ravikumar and S. Chopra. "Testing interconnects in a system chip", In *Proc. Int. Conf. VLSI Design*, pp. 388-391, 2000.

[48] K. Sekar and S. Dey, "LI-BIST: A low-cost self-test scheme for SoC logic cores and interconnects", In *IEEE VLSI Test Symp*, pp. 417-422, 2002.

[49] R. Pendukar, A. Chatterjee and Y. Zorian. "Switching activity generation with automated BIST synthesis for performance testing of interconnects", In *IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS*, vol. 20, no. 9, pp. 1143-1158, Sept. 2001.

[50] Y. Zhao , S. Dey , L. Chen. "Double sampling data checking technique: an online testing solution for multisource noise-induced errors on on-chip interconnects and buses", In *IEEE Trans on VLSI Systems*, vol. 12, no. 7, p. 746-755, July 2004.

[51] A. Attarha, M. Nourani. "Testing Interconnects for Noise and Skew in Gigahertz SoC", In *Proc. of Int. Test Conf.*, pp. 305-314, 2001.

[52] C.Su, Y.T.Chen, M.J.Huang, G.N.Chen and C.L.Lee. "All Digital Built-in Delay and Crosstalk Measurement for On-Chip Buses", In *Proc. of DATE Conf.*, pp. 527-531, March 2000.

[53] Ulf Pillkahn. "Structural test in a board self test environment", In *Proc. IEEE Int. Test Conf. (ITC2000)*, pp. 1005-1012, Oct 2000.

[54] C.-A. Chen, S.K. Gupta. "BIST/DFT for performance testing of bare dies and MCMs", In *Proc. of Electro94 International Conf.*, pp. 803-812, May 1994.

[55] X. Bai, S. Dey, and J. Rajski. "Self-test methodology for at-speed test of crosstalk in chip interconnects", In *Proc. Design Automation Conf.*, pp. 619224, 2000.

[56] C. Grecu, P. Pande, A. Ivanov and R. Saleh. "BIST for network-on-chip interconnect infrastructures", In *Proc. 24th IEEE VLSI Test Symposium*, pp. 30-35, 2006.

[57] T. Bengtsson, A. Jutman, S. Kumar, R. Ubar, Z. Peng. "Offline testing of delay faults in NoC interconnects", In *Proceeding of the 9th EUROMICRO Conference on Digital System Design*, pp. 677-680, IEEE 2006.

[58] Marinissen, E.J.; et al. "On IEEE P1500's Standard for Embedded Core Test", In *Journal of Electronic Testing: Theory and Applications*, vol. 18, pp. 365-383, 2002.

[59] X. Tran, Y. Thonnart, J. Durupt, F. Bertrand, V. Beroulle and C. Robach. "A Design-for-Test Implementation of an Asynchronous Network-on-Chip Architecture and its Associated Test Pattern Generation and Application", In *Second ACM/IEEE International Symposium on Networks-on-Chip*, pp. 149-158, IEEE 2006.

[60] Y. Zorian, E. Marinissen and S. Dey. "Testing Embedded-Core Based System Chips", In *Proceedings of the 1998 IEEE International Test Conference*, pp. 130-143, 1998.

[61] "IEEE 1149.1-1990, Standard test access port and boundary scan", *Atmel*, Available: www.Atmel.com.

[62] Aktouf, C. "A Complete Strategy for Testing an on-chip Multiprocessor Architecture". In *IEEE Design Test of Computers*, vol. 19, no. 1, pp. 18-28, 2002.

[63] C. Liu, Z. Link and D.K. Pradhan. "Reuse-based test access and integrated test scheduling for Network-on-Chip", In *Proc. Design, Automation and Test in Europe*, vol. 1, pp. 303-308, March 2006.

[64] Wu, Y. and MacDonald, P. "Testing ASICs with Multiple Identical Cores", In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 3, pp. 327-336, 2003.

[65] M. Hosseinabadi, A. Banaiyan, M. N. Bojnordi, and Z. Navabi. "A Concurrent Testing Method for NoC Switches" In *Proc. Design Automation and Test in Europe(DATE)*, vol. 1, pp. 1171-1176, 2006.

[66] C. Grecu, P. Pande, B. Wang, A. Ivanov and R. Saleh. "Methodologies and Algorithms for Testing Switch-Based NoC Interconnects", In *Proceedings of the 2005 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT05)*, pp. 238- 246, IEEE 2005.

[67] A.Radulescu, J.Dielissen, S.Gonzlez Pestana, O.Prakash Gangwal,E.Rijpkema, P.Wielage and K.Goossens. "An Efficient On-Chip NI Offering Guaranteed Services,Shared-Memory Abstraction, and Flexible Network Configuration", in *IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS* , vol. 24, no. 1, pp. 4-17, Jan 2005.

Appendices

Test block control signal figures

A

- In Figure A.1, the way test block and light NI are communicating via handshake signals is depicted. Whenever light NI has a test message ready to be sent to test block, the valid-test-data signal becomes active, and when ever the test block accepts the test data, it activates accept-test-data in response. When a test response message is ready to be sent to light NI to get packetized, the test block activates the valid-test-response signal and when light NI accepts the message it activates the accept-test-response in response.

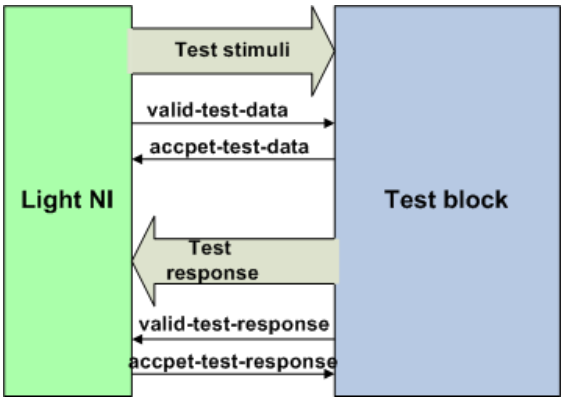


Figure A.1: Test block interacting with light NI

- Figure A.2 shows the data and control lines between test block and input register. Test block By activating different control signals (serial-out, load-in-inputreg) at different phases of test, produces parallel or serial output from the input register.
- Figure A.3, depicts the data and control lines between test block and output register. By activating control signals by test block the input of output register can be either parallel or serial. This depends on different phases of test.
- In Figure A.4, two control signals sent by the test block to router-under-test. *Hold* signal is used to hold the clock of router and *scan-enable* signal is used to enable the scan mode of internal flip-flops of router and create the scan chains in the router.

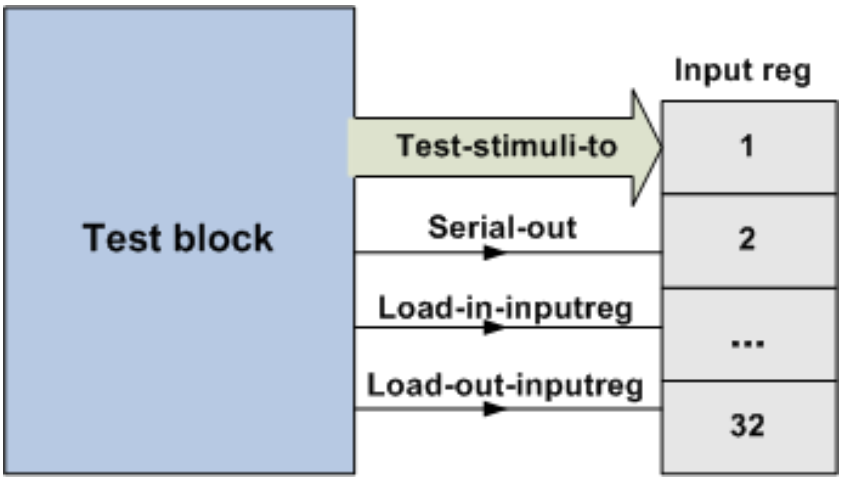


Figure A.2: Test block interacting with input register

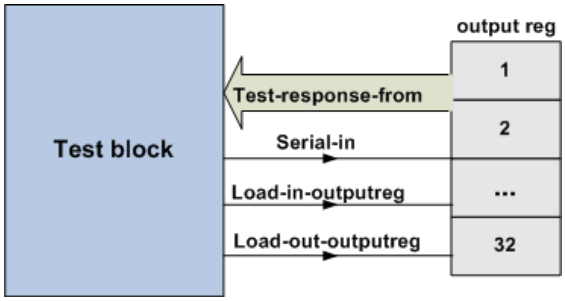


Figure A.3: Test block interacting with output register

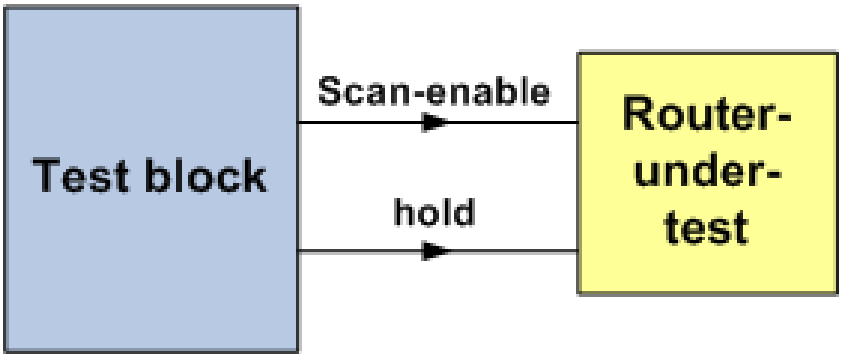


Figure A.4: Test block interacting with router-under-test

state transition figures

B

- Figure B.1 shows the situation in test block that the state machine is in scan-in state (S2) and the scan chain in full. At this point if the test pattern words that should be scanned-in to scan chain are completed, the state machine enters to state S4 (scan-out state). Otherwise, if test pattern is not completed yet, the state machine enters to S7, which waits to receive the next test pattern word in input register and then enters S4.

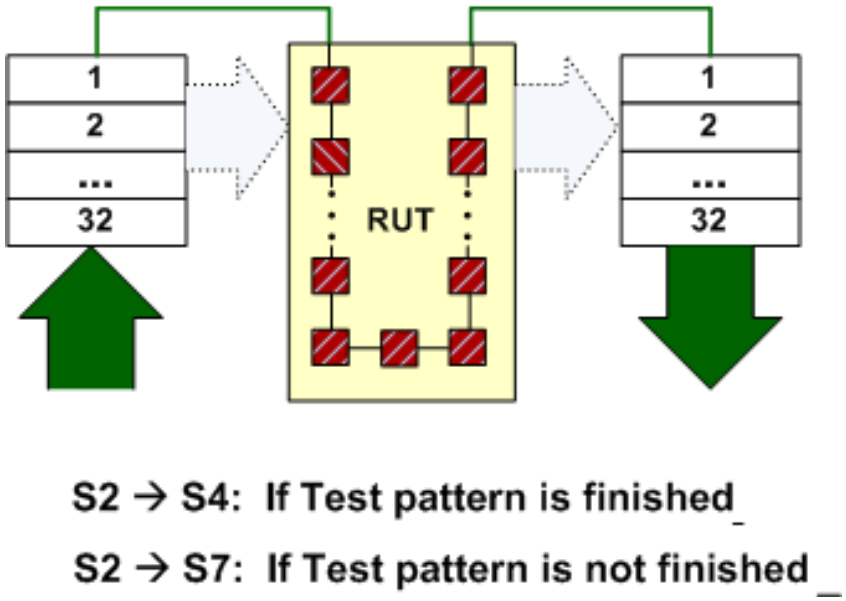


Figure B.1: Transition between S2 and S4 or S7

- Figure B.2 depicts situation that state machine is in state S2, while the scan chain is not full yet. At this situation if the test pattern is completed the state machine has to enter S6. In S6 the contents of scan chain will be shifted toward the end of scan chain. If the test pattern is not completed yet, the state machine enters state S3. In S3, it waits to receive the next word of test pattern.
- Figure B.3 illustrate a situation that the state machine is in scan-out state (S4), and all input register, output register and scan chain are full. In this situation, if the test pattern in finished, the next state will be state S5. In S5

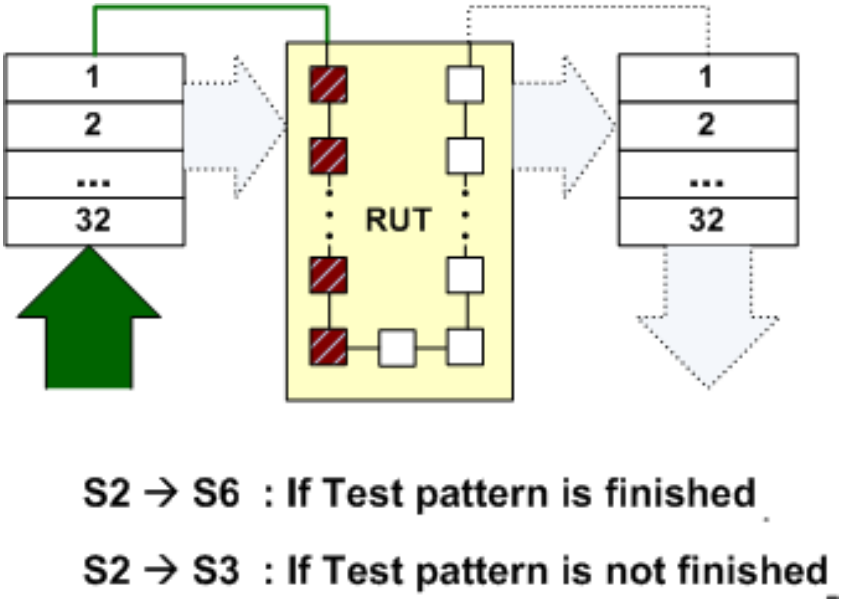


Figure B.2: Transition between S2 and S3 or S6

the state machine waits for light NI to receive the test response word that is in output register, so the output register becomes empty and the rest of test response get loaded out. However, if the test pattern is not finished yet, the state machine will enter state S7.

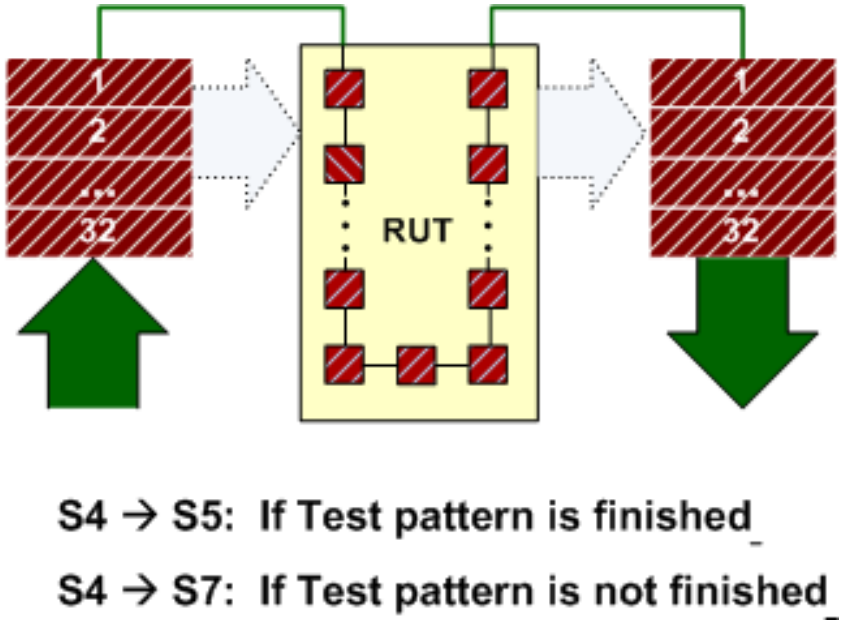


Figure B.3: Transition between S4 and S5 or S7

- Figure B.4 shows, the second phase of testing the router. The scan chain of the router is full which means the internal flip-flops of the router are initialized. When the test pattern word is received, the state machine will transit to state S12. In S12 the parallel data will be applied to functional input of the router.

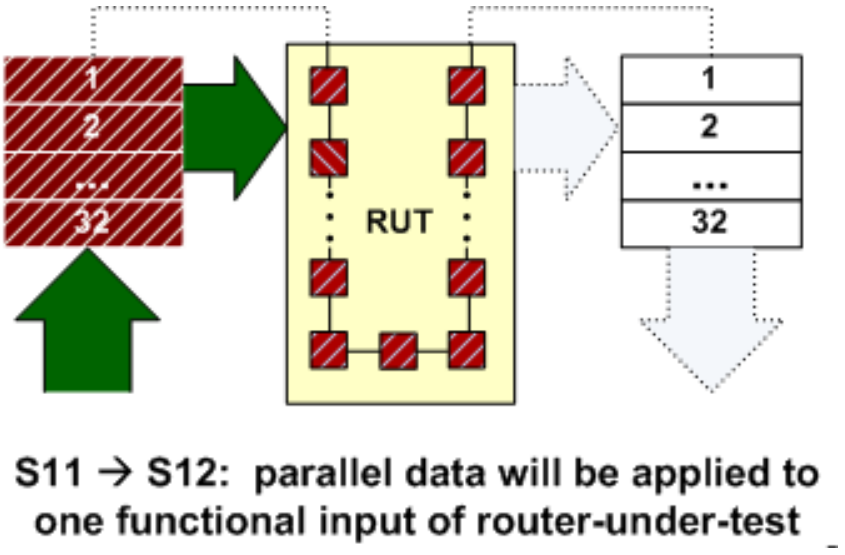


Figure B.4: Transition between S11 and S12

- In Figure B.5, state machine is in state S12. After one clock cycle that the test data is applied to the combinational part of the router, the test response will be ready in one of the functional outputs of the router and stored in output register.

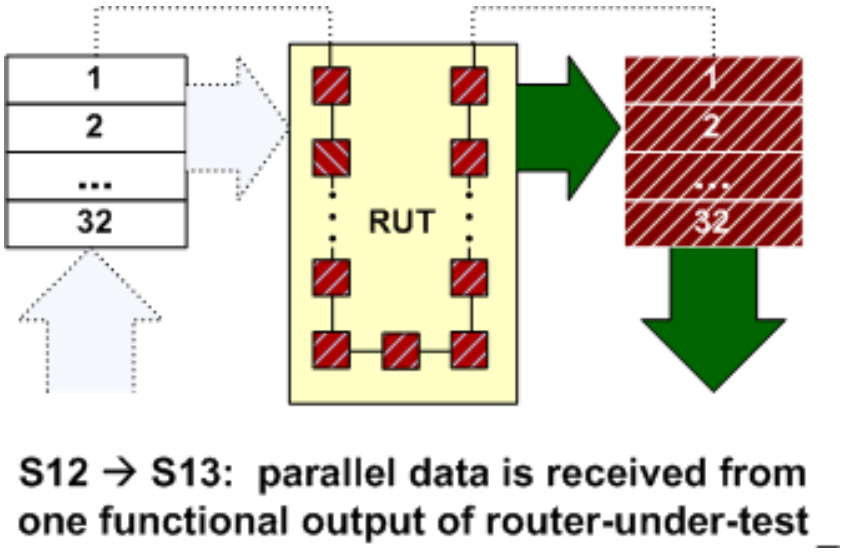


Figure B.5: Transition between S12 and S13