Offline Test and Diagnosis of Network-on-Chips

Haupt-Seminar

June 09, 2009

Author: Tarek H. Zaki
Supervisor: Dipl.-Inf. Stefan Holst
Reviewers: Prof. Dr. M. Radetzki, Prof. Dr. H.-J. Wunderlich
Contents

List of Figures ii
List of Tables iii
Nomenclature v
1 Motivation 1

2 Testing Inter-Switch Wire Segments 3
2.1 Maximum Aggressor Fault Model . . . . . . . . . . . . . . . . 3
2.2 The Built-In-Self-Test Structure . . . . . . . . . . . . . . . . 4

3 Testing Switch Blocks 7
3.1 Switch Architecture . . . . . . . . . . . . . . . . . . . . . . . 7
3.2 Testing the FIFO Buffers . . . . . . . . . . . . . . . . . . . . 7
3.3 Testing the Routing Logic Blocks . . . . . . . . . . . . . . . . 9
3.4 Testing the Multiplexers and Intra-Switch Links . . . . . . . 14

4 Overall Test Strategy 20

5 Conclusion 22

Bibliography 23
List of Figures

1.1 Portion of the NoC interconnect architecture . . . . . . . . . . 1
2.1 Possible crosstalk errors under the MAF model . . . . . . . . 4
2.2 Block diagram of the BIST for the inter-switch link . . . . . 5
2.3 Test sequence for testing the inter-switch links . . . . . . 6
3.1 The general switch architecture . . . . . . . . . . . . . . . . 8
3.2 Shared BIST controller for testing the FIFOs . . . . . . . . . 8
3.3 External test structure for RLBs . . . . . . . . . . . . . . . . 14
3.4 Timing diagram of broadcasting test vectors . . . . . . . . . 14
3.5 Two different NoC topologies with optimum broadcast routing 15
3.6 Targeted structural faults for the MUX . . . . . . . . . . . . 15
3.7 Test configurations for testing MUXs in NoCs . . . . . . . . 16
3.8 Input/Output ports of a switch . . . . . . . . . . . . . . . . 17
3.9 Flowchart of the diagnosing algorithm for faulty link . . . . . 18
3.10 Example to diagnose faulty intra-switch link . . . . . . . . 19
4.1 Composite test strategy for testing NoCs . . . . . . . . . . . . 21
List of Tables

2.1 Test area overhead of the TDG and TED blocks . . . . . . . . 6
3.1 FIFO buffer fault models . . . . . . . . . . . . . . . . . . . . 10
3.2 The effects of each FIFO buffer fault model . . . . . . . . . 11
3.3 Composite functional test pattern for FIFO buffers . . . . . . 13
3.4 Classes of faulty intra-switch links in a switch . . . . . . . . 18
Nomenclature

BF  Bridging Fault, page 9
BIST  Built-In-Self-Test, page 2
Del  Defined time units for the DRF to occure, page 9
DRF  Data Retention Fault, page 9
DSM  Deep-SubMicron, page 3
DV  Data Valid, page 9
FIFO  First-In-First-Out Buffer, page 2
IP  Intellectual Property, page 1
LRA  Local Response Analyzer, page 9
MAF  Maximum Aggressor Fault, page 3
MUX  Multiplexer, page 2
NoC  Network-on-Chip, page 1
RD  Read Data, page 9
RLB  Routing Logic Block, page 2
RNI  Resource Network Interface, page 1
RS  Reset, page 9
SA  Stuck-at Fault, page 9
SW  Switch, page 1
$T_{\text{FIFO}}$  Test latency for the FIFO buffers, page 9
NOMENCLATURE

TAM Test Access Mechanism, page 2
TAS Test Access Switch, page 12
TDG Test Data Generator, page 4
TED Test Error Detector, page 4
TF Transition Fault, page 9
WD Write Data, page 9
Chapter 1

Motivation

Network-on-Chip (NoC) design methodology has lately emerged to replace the traditional bus based architectures [Kuma05]. This new paradigm allows us to integrate multiple interconnected processors on chip to provide higher bandwidth, lower latency and scalable communication infrastructure. Referring to Fig. 1.1 NoC is defined to be set of switches (SWs) connected by point-to-point channel medium and resource network interfaces (RNI) to connect the processing IP cores with the corresponding switches. Different topologies for NoCs were proposed in literature like mesh-based (grid), torus, hypercube, ring and fat-tree. The paradigm is based on packet-switched network or message-passing communication model for sending and receiving messages between the switches.

![Figure 1.1: Portion of the Network-on-Chip (NoC) interconnect architecture.](image)

Testing the NoC-based systems is mainly concerned with the testing of the interconnect infrastructure including the switches and the inter-switch wire segments, and testing the functional IP cores and their corresponding resource network interfaces [Pand05, Akto02]. The main target of this paper
1. Motivation

is to discuss the developed testing schemes for testing separately all the basic blocks of a NoC while minimizing the testing area overhead and time. Moreover, a composite test sequence is proposed which combines the testing schemes to sequentially test and diagnose all the interconnected blocks of a NoC.

There are mainly two different alternatives for testing NoC; Built-In-Self-Test (BIST) and external test methodologies [Raik06]. The main advantage of the BIST strategy is the lower test time if it is able to test almost all the components simultaneously. However, it has relatively large area overhead. On the other hand, the external test configuration usually reuses the NoC infrastructure itself for test instead of using dedicated test bus as a test access mechanism (TAM) [Cota04, Mari98, Varm98]. This strategy shows a significant advantage on low pin and area overhead. The expense of this significant low pin and area overhead is higher test time. A technique called multicast is discussed in this paper which inherits parallelism to reduce the testing time in this external test strategy. Therefore, there is always a tradeoff between test area overhead and time where both should be minimized as much as possible. For every presented test method in this paper, the corresponding fault model as well as the fault coverage is discussed.

Testing the interconnect infrastructure should address two main targets; the inter-switch wire segments and the switches. First, Chapter 2 discusses a BIST scheme for testing the inter-switch wire segments which pose a challenge due to the poor controllability and observability through the NoC. This is because the links are embedded and spread across the entire chip [Grec07]. Second, the switches are composed of three main blocks; the First-In-First-Out (FIFO) buffers, Routing Logic Block (RLB) and Multiplexers (MUXs) [Pand05]. Chapter 3 provides different testing methods for testing and diagnosing separately all the main components of the switches. Third, testing the functional IP cores and their corresponding resource network interfaces is not discussed in this paper since it can be tested by the same external test technique used to test RLBs for the switches. The only difference will be in the test vectors broadcasted through the network [Cot03, Cota03].

After discussing the several techniques and alternatives for testing separately the different components of a NoC, in Chapter 4 a composite test sequence is proposed which chooses between the different alternatives and utilizes them in a specific order. This is to integrate the test and diagnosis of all the components of a NoC and to take into consideration the interdependency between the testing schemes.
Chapter 2

Testing Inter-Switch Wire Segments

2.1 Maximum Aggressor Fault Model

In this chapter, a Built-In-Self-Test (BIST) methodology is discussed for testing the inter-switch links of the NoC [Grec06, Grec07]. The inter-switch interconnect links are the wire segments that create the medium for intelligent switches to communicate through. This test methodology reuses the NoC infrastructure itself to transport test data instead of using a dedicated Test Access Mechanism (TAM) [Cota04].

Inter-switch links are exposed mainly to crosstalk errors during data transfer between the switches. They are affected by such type of error to the fact that inter-switch links are the longest global wires other than the power and the clock lines. The fault model that is taken into account in this test strategy is specific for the deep-submicron (DSM) technologies. This fault model is proposed by Cuviello et al. for global interconnects and referred as Maximum Aggressor Fault (MAF) [Cuvi99]. This fault model addresses the worst case, where a line (victim line) is affected through cross-talk by simultaneous transitions on all other lines (aggressor lines). Therefore, for N inter-switch lines, the worst case situation when one victim line is affected by N-1 aggressor lines. This fault model covers a wide range of defects; design errors, violation of design rules, process variations, stuck-at, stuck-open, bringing faults and physical defects. There are six crosstalk errors addressed by this fault model; positive/negative glitch, rising/falling delay and rising/falling speed-up. For two aggressor lines (Y1 and Y3) and one victim line (Y2), the six possible crosstalk errors are shown in Fig. 2.1. The dotted output signals are the expected fault-free signal while the bold output signals are the unexpected outputs due to the effect of MAF. Speed-up
2. Testing Inter-Switch Wire Segments

occurs when the aggressor lines have the same transitions as the victim line. On the other hand, the delay occurs when the aggressor lines have opposite transition to the victim line. As for the glitches, they occur when there are transitions on the aggressor lines while the victim line is stable. For each crosstalk error, 2 test vectors are needed. Therefore, for N inter-switch lines, 6N*2 test vectors are required to apply this test strategy.

![Diagram of crosstalk errors]

Figure 2.1: Possible crosstalk errors under MAF model.

2.2 The Built-In-Self-Test Structure

Fig. 2.2 illustrates the block diagram of the BIST methodology for one inter-switch link which is proposed in [Grec06]. The block diagram is composed of two main blocks; the test data generator (TDG) and the test error detector (TED). The role of the TDG is to generate or produce the test sequences for testing the six crosstalk possible errors explained above. These test vectors are then propagated through the line under test. Finally, the data on the lines are then sampled and compared by the TED circuit to ensure consistency with the expected results. The TDG and TED are placed before the driver and after the loads respectively. This is to take into account any undesired errors occurred at the large size drivers or loads. Here it is assumed that the transaction from the source to the destination through the inter-switch lines requires only one clock cycle. Therefore, the TED simply also generates internally the same test vector sequences as the TDG but delayed.
2. Testing Inter-Switch Wire Segments

by one cycle. This is to ensure consistency between the sampled data after
the loads and the generated data internally in the TED for comparison. The
self test structure differs slightly depending on whether the links uses unidi-
rectional or bidirectional transactions. For the unidirectional transactions,
only TDG/TED pair is sufficient for each link. However, the bidirectional
transactions require two pairs for the test to be conducted in both directions.

Figure 2.2: Block diagram of the BIST for the inter-switch link.

As mentioned before, 6N*2 test vectors are required for testing the six
possible crosstalk errors. Though, the test vectors can be merged to reduce
the test time. After merging, the number of test vectors are reduced from
12N to 8N. Fig. 2.3 shows a possible test sequence and the state diagram for
the TDG/TED. As shown for the 8 test vectors, the six crosstalk errors are
still covered in the test. Note that ‘i’ is indication for the victim line which
is under test and the other 4 lines are aggressor lines. The TDG consists
of three main blocks; a finite state machine, a counter and a barrel shifter.
The finite state machine is implemented to generate the required test vector
every cycle. This is achieved by implementing the state diagram that is
shown in Fig. 2.3(b). As for the counter, it controls the barrel shifter to
select the desired victim line under test. Each victim line requires 8 clock
cycles. Therefore, for N inter-switch lines, a total of 8N clock cycles are
required for fully testing all the lines. The time needed for testing a single
link on the NoC is the same needed for testing arbitrary number of links.
This is because all the links are tested simultaneously in this test strategy.

If only subset of the crosstalk errors is of interest, the state machine will
slightly differ to include only the desired subset tests. The implementation
of the state machine for the TED is almost similar to the implementation of
the TDG; the only difference is that the state machine in the TED is delayed
by one clock cycle. This is to ensure the test vector being transmitted over
the link is consistent with the internally generated test vectors in the TED. Additional XOR network is implemented in the TED to compare the locally generated test vector sequence with the sampled line values. Any mismatch will raise an error flag indicating error detection. If the links requires multiple clock cycle for transmitting the test vector, the state machine of the TED will include additional wait states between the test states to ensure consistency of the operation.

From the first sight of the operation, it seems that the area overhead is large since a pair of TDG/TED is required for each inter-switch link. However, by comparing the area of the TDG and TED blocks implemented using 2-input NAND gates shown in Table 2.1 with the typical switch size which is around 30K gates [Grec06, Pand03]. It results in a very low area overhead for this BIST methodology.

![Diagram](image)

Figure 2.3: Test sequence for testing the inter-switch line i and the corresponding state machine.

<table>
<thead>
<tr>
<th>Line width (bits)</th>
<th>TDG (gates)</th>
<th>TED (gates)</th>
<th>Area overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>287</td>
<td>314</td>
<td>2.0%</td>
</tr>
<tr>
<td>32</td>
<td>471</td>
<td>506</td>
<td>2.3%</td>
</tr>
<tr>
<td>64</td>
<td>812</td>
<td>866</td>
<td>5.6%</td>
</tr>
</tbody>
</table>

Table 2.1: Test area overhead of the TDG and TED blocks.
Chapter 3

Testing Switch Blocks

3.1 Switch Architecture

Referring to Fig. 3.1, the general switch architecture consists of three main components; the First-In-First-Out Buffers (FIFOs), Routing Logic Block (RLB) and Multiplexers (MUXs) [Raik07, Raik06]. The role of the FIFO Buffers is to receive and store the incoming packets coming from the neighbor switches. Each packet is composed of header, payload, and trailer flow control units (flits) [Hoss06]. The header flit is decoded by the routing logic block (RLB) of the switch. Based on the target address information in the head, the subsequent flits are forwarded through the MUXs to one of the output ports of the switch. This technique simply establishes a pipelined path from the source to the destination through number of intra switch checkpoints. Better design of the switches will establish shorter paths which result in less transversal time. Testing these switches should cover all the three main components. So in the following sections, different techniques for testing the three components separately will be discussed.

3.2 Testing the FIFO Buffers

Testing the FIFO buffers is considerably a hard task. Since the buffers are spread all over the chip using relatively large area. Accordingly, the probabilities of faults or defects are significantly higher for the buffers compared to the other components in the switch.

A concurrent test technique is proposed in [Grec05]. As shown in Fig. 3.2(a), a shared Built-In-Self-Test (BIST) controller is used for testing all the FIFO buffers at a time instead of having a dedicated BIST for each buffer. Regarding the test, this results in an acceptable area overhead. The testing
Figure 3.1: The general switch architecture.

Figure 3.2: (a) Shared BIST controller for testing the FIFOs, (b) Dual port FIFO structure.
3. Testing Switch Blocks

scenario is as follows, the BIST controller writes the same test packets to all the FIFO buffers on the chip. Then the Local Response Analyzer (LRA) of each buffer will detect and report back if there is an error in the buffer. Diagnosing the faults in this case would be simple because each LRA is reporting for a unique buffer after a specific written test input packet.

Referring to block diagram of the FIFO buffer shown in Fig. 3.2(b), all the possible fault models are summarized in Table 3.1. Goor, Schanstra and Zorian proposed in [Goor95] a composite functional test pattern which covers all these fault models with a complexity of O(n); where n is the buffer size. Therefore, the BIST controller and the LRA can be designed to apply this test pattern. In order to understand how this test pattern detects all these fault models, Table 3.2 illustrates the effects of each fault. The test pattern and the corresponding detected faults are shown in Table 3.3. Noting that, $\uparrow^n{0}$ denotes that sequence of operations are performed from memory cell 0 to cell n-1. Moreover, $\uparrow^n{G}G$ denotes that the alternating values $G$ and $G$ are written sequentially to adjacent memory cells. The full flag (FF) and empty flag (EF) are regularly checked after every step by the LRA for error detection. If the values of checked flags don’t match the expected values, the LRA will report back for an error in the corresponding FIFO buffer.

Additional fault model can be taken into consideration which is the dual port coupling fault [Grec05]; where a write operation on a cell might cause an erroneous read operation on the adjacent cell and vice versa. To cover the Bridging Faults (BFs) also in this test, four test patterns are used 0101, 1010, 0000, 1111. For each test pattern the following test sequence is applied:

$$w\{ \uparrow^{n-1} (wr) \} r$$ (3.1)

The first write initializes the FIFO. Then (n-1) read/write operations are applied simultaneously on all adjacent cells. The last read is to reset the FIFO again for the next test pattern. 4(n+1) test cycles are required for this test. Therefore, the total test latency for all the FIFO buffers is the sum of all testing times which remains linearly dependent on the buffer size.

$$T_{FIFO} = (2n + 3Del + 4) + 4(n + 1) = 6n + 8 + 3Del$$ (3.2)

3.3 Testing the Routing Logic Blocks

After testing the inter-switch wire segments and the FIFO buffers, the on-chip network including the FIFO buffers can be used to broadcast test vectors
Table 3.1: FIFO buffer fault models.

<table>
<thead>
<tr>
<th>Memory array fault model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Stuck-at Fault SA</td>
<td>After any operation the memory cell remains 0 or 1.</td>
</tr>
<tr>
<td>2. Transition Fault TF</td>
<td>After a transition operation to 1 the memory cell remains 0 or vice versa.</td>
</tr>
<tr>
<td>3. Data Retention Fault DRF</td>
<td>After a defined time “Del” the memory cell loses its data.</td>
</tr>
<tr>
<td>4. Bridging Fault BF</td>
<td>Short connection between two bitlines, most probably adjacent bitlines.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Addressing fault model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5. SA0 fault on $RD/WD_i$</td>
<td>Stuck at 0 fault on a read/write from the read/write control units.</td>
</tr>
<tr>
<td>6. SA1 fault on $RD/WD_i$</td>
<td>Stuck at 1 fault on a read/write from the read/write control units.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FIFO logic fault model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7. SA0 fault on $DV_i$</td>
<td>Data valid signal indicates that memory cell $i$ is always empty.</td>
</tr>
<tr>
<td>8. SA1 fault on $DV_i$</td>
<td>Data valid signal indicates that memory cell $i$ is always full.</td>
</tr>
<tr>
<td>9. DRF0 fault on $DV_i$</td>
<td>Data valid signal will become 0 after time “Del”.</td>
</tr>
<tr>
<td>10. DRF1 fault on $DV_i$</td>
<td>Data valid signal will become 1 after time “Del”.</td>
</tr>
<tr>
<td>11. Improper Reset RS</td>
<td>The data valid signals for all memory cells are not initialized after the reset.</td>
</tr>
<tr>
<td>12. Improper write WO</td>
<td>If a data can be written to the memory while it is full, or if the data is not written properly.</td>
</tr>
<tr>
<td>13. Improper Shift</td>
<td>The data is not shifted properly in the memory cell array.</td>
</tr>
<tr>
<td>14. Improper Read RO</td>
<td>If a data can be read from the memory while it is empty, or if the data is not read properly.</td>
</tr>
</tbody>
</table>
Table 3.2: The effects of each FIFO buffer fault model.

<table>
<thead>
<tr>
<th>Fault</th>
<th>Fault effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faults 1-4</td>
<td>The effects of these faults is that the entire memory cells will change their values. To test these faults, write a specific pattern to the memory array and read them again. If a different value is read from the FIFO, this implies that one of the 4 faults occured. The 4 faults can be differentiated easily from the unexpected read value.</td>
</tr>
<tr>
<td>Fault 5</td>
<td>This fault prevents the cell from being read/written, but it won’t affect the data valid signal of this cell. Therefore, this fault has the same effect like Faults 1-4 where an unexpected value will be transported to the output.</td>
</tr>
<tr>
<td>Fault 6</td>
<td>This fault will lead to read/write from/to cell i every cycle. But as Fault 5, this fault won’t affect the data valid signal of this cell. So the unexpected initially stored value in memory cell i will propagate to the output like Faults 1-5. Therefore, Faults 1-6 can be tested by the same method.</td>
</tr>
<tr>
<td>Fault 7</td>
<td>The effect of this fault is that no data transfers can take place from the memory cell i, so there will be no shifting down of the data. This implies that any number of writes can be done into the FIFO and no word can be read from the FIFO. Therefore, EF will always be 1.</td>
</tr>
<tr>
<td>Fault 8</td>
<td>The effect of this fault is that no data transfers can take place into the memory cell i. This implies that less than n words can be written into the FIFO.</td>
</tr>
<tr>
<td>Fault 9</td>
<td>The data in memory cell i will be lost after waiting $Del$ time units. To test this fault, wait $Del$ time units after filling the FIFO. Then less than n words can be read from the FIFO.</td>
</tr>
<tr>
<td>Fault 10</td>
<td>The invalid data in memory cell i will be valid after waiting $Del$ time units. To test this fault, wait $Del$ time units after ensuring that the FIFO is empty. Then the EF will flip from 1 to 0 indicating a valid undesired data in the FIFO.</td>
</tr>
<tr>
<td>Fault 11</td>
<td>The effect of improper reset operation is that the EF will not indicate that FIFO is not empty by having the value 0, so one or more words can be read from the FIFO and less than n words can be written into the FIFO.</td>
</tr>
</tbody>
</table>
### 3. Testing Switch Blocks

<table>
<thead>
<tr>
<th>Fault</th>
<th>Fault effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault 12</td>
<td>The effect of this fault is one of the following:</td>
</tr>
<tr>
<td></td>
<td>- Data is written to the FIFO while its full FF=1. To test this fault, after filling the buffer, write a value that is different from the values in the buffer. Then read all the FIFO to see if the data was written while it was full or not.</td>
</tr>
<tr>
<td></td>
<td>- The value written to the buffer is different from the expected value. This fault behaves like Fault 1 where the memory cell 0 is stuck with an unexpected value, so the unexpected value stored in this memory cell will detected at the output.</td>
</tr>
<tr>
<td></td>
<td>- After a write operation, the data valid signal remains 0 for memory cell 0. This is exactly as Fault 7 where DV[0] is stuck at 0.</td>
</tr>
<tr>
<td>Fault 13</td>
<td>The effects of this fault is similar to fault 1, 7 or 8.</td>
</tr>
<tr>
<td>Fault 14</td>
<td>The effect of this fault is one of the following:</td>
</tr>
<tr>
<td></td>
<td>- Read is allowed while the FIFO is empty. This effect is similar to the effect of Fault 10.</td>
</tr>
<tr>
<td></td>
<td>- Incorrect value read from the FIFO. This effect is also similar to the effect of Fault 1.</td>
</tr>
</tbody>
</table>

from an external test source to test the combinational routing logic blocks. There are two possible methods for this external test; unicast sequential test and multicast parallel test [Hoss06, Grec05]. In this section, only the multicast parallel test is discussed, as it provides a lower total testing time.

This technique requires that all switches are similar. In this case for the same test vector, all switches should provide the same output. Therefore, the idea of this method is to broadcast the test vectors through the network to all switches. An external test source is used to transmit those test vectors to the network. The switches which are directly connected to this test source are called Test Access Switches (TAS). Each switch forwards the test vectors in addition to its primary results to the neighbor switches for comparing them with their outputs. In this way, any mismatch between the expected output values and the primary output values will result in fault detection. In case of
3. Testing Switch Blocks

Table 3.3: Composite functional test pattern for FIFO buffers.

<table>
<thead>
<tr>
<th>Step</th>
<th>Operation</th>
<th>Expected flags</th>
<th>Operations</th>
<th>Faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td></td>
<td>FF EF</td>
<td>count</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>RS</td>
<td>0 1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>2a</td>
<td>↑n-2 wG</td>
<td>0 0</td>
<td>n-1</td>
<td>8, 11</td>
</tr>
<tr>
<td>2b</td>
<td>wG</td>
<td>1 0</td>
<td>1</td>
<td>7, 8, 11</td>
</tr>
<tr>
<td>3</td>
<td>wait</td>
<td>1 0</td>
<td>Del</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>wG</td>
<td>1 0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>rG</td>
<td>0 0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>wG</td>
<td>1 0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>wait</td>
<td>1 0</td>
<td>Del</td>
<td></td>
</tr>
<tr>
<td>8a</td>
<td>↑n-2 rG</td>
<td>0 0</td>
<td>n-1</td>
<td>1-6, 12a</td>
</tr>
<tr>
<td>8b</td>
<td>rG</td>
<td>0 1</td>
<td>1</td>
<td>1-6</td>
</tr>
<tr>
<td>9</td>
<td>wait</td>
<td>0 1</td>
<td>Del</td>
<td>10</td>
</tr>
</tbody>
</table>

Fault detection, a flag is transmitted back until it reaches the TAS to notify the external test source the existence of the fault.

To gain more insight of this method, Fig. 3.3 illustrates how the switches are connected to the test source. Moreover, Fig. 3.4 clarifies how the packets are broadcasted between the switches. Noting that a capture cycle (cap) is needed by the switch to compute its primary output.

If the basic packet routing technique (flooding) is used in this multicast method, the header size will increase and will result in a high overload. Therefore, another header encoding technique is known as bitstring is proposed in [Siva00] which solves this problem. Another problem of this multicast method is to locate the topological center of the network to connect the test source. The topological center is defined to be the switch with a minimum distance from other switches [Hoss06]. Dijkstra’s algorithm can be used to locate this topological center [Corm90]. Moreover, Reverse path forwarding algorithm proposed in [Dala78] can be used to find the optimum broadcast routing. This algorithm defines how the packets will be broadcasted through the network. It configures each switch port to be a master, slave or an idle port to form the minimum broadcast tree. This external test method is independent of the NoC topology. Fig. 3.5 shows two examples for different NoC topologies to clarify the configured minimum broadcast tree where each switch receives the test vectors from the master port and forwards the results to all the slave ports. The dotted lines in the figure represent the unused channels in the test mode. Noting that other
3. Testing Switch Blocks

![Diagram of external test structure for RLBs.](image)

Figure 3.3: External test structure for RLBs.

![Timing diagram of broadcasting test vectors through the NoC of Fig. 3.3.](image)

Figure 3.4: Timing diagram of broadcasting test vectors through the NoC of Fig. 3.3.

tree configurations in these two networks might lead to a larger depth of the tree and longer testing time. As a result of these two algorithms, the total testing time of this multicast method is significantly reduced.

3.4 Testing the Multiplexers and Intra-Switch Links

As shown previously in the general switch architecture, crossbar multiplexers are used at each port of the switch to select between the other 4 ports for sending the packets through it. Testing the crossbar multiplexers pose three main targets; the 4-input signals, the select signal and the output signal. Referring to Fig. 3.6 each input signal will be tested at a time. This is done by ensuring a signal value at the tested input different from the other
3. Testing Switch Blocks

3-inputs. In this way, the path will be sensitized at the output and any undesired fault in the multiplexer will result in an unexpected output. This technique covers all the three targets required for testing the multiplexers. Note that all possible address values have to be applied to fully test the multiplexers.

![Figure 3.5: Two different NoC topologies with optimum broadcast routing.](image)

![Figure 3.6: Targeted structural faults for the MUX.](image)

An external test approach for mesh-based network is proposed in [Raik06, Raik07]. In this testing scheme three test configurations are used to test all the possible routings for the cross-bar multiplexers. The first configuration covers all the straight path routings (north-south, south-north, east-west and west-east). Fig. 3.7(a) illustrates how the north-south straight path routings are tested in this configuration. If no fault detected in the first configuration, then straight path routings are proved to be non-faulty. After the first configuration, the second test configuration shown in Fig. 3.7(b),(c)
3. Testing Switch Blocks

is applied, it covers all the turning path deterministic routings (XY-routing). In this configuration only one diagonal is shown in the figures. However, the test is done for all possible diagonals in the network. If the network supports also deterministic YX-routings, then the same test configuration is applied. The third and the last configuration shown in Fig. 3.7(d) is done thereafter, it covers the resource connections with all other ports. This last test is applied by writing then reading the written value to the resources for all rows in the network. In Fig. 3.7(d), only writing to the resource from the north is illustrated. However, this test is applied by writing to and reading from the resources from all possible directions.

The idea of this test is to send pipelined test sequences ensuring that each target multiplexer connection will have a discriminated value from the other connections. By observing the examples for the three test configurations shown in Fig. 3.7 on each switch the tested route has a distinguished signal value compared to the other routes. Note that the dark and the light arrows represent the signal value $X$ and $\overline{X}$ respectively. By this approach it is ensured that the value at the selected multiplexer input is differentiated from the values at the other inputs.

![Figure 3.7: Test configurations for testing MUXs in NoCs: (a) Straight path routing, (b) XY-routing, (c) opposite XY-routing, (d) resource routing.](image-url)
Testing results shows that this external testing method provides near to 100% structural fault coverage for the crossbar multiplexers (stuck-at, transition, open and short faults) with complexity proportional to the square root of the number of switches \([\text{Raik07}]\). Therefore, this approach is a practical alternative to the BIST based approaches, as it provides a high test quality but with lower area overhead.

After testing the multiplexers in the switches, it is interesting to locate the faulty switches in the network. Referring to Fig. 3.8 there are 20 different intra-switch links inside each switch for the non-bouncing switches (no feedback links). Note the difference between intra-switch and inter-switch links; the inter-switch links are the links connecting the switches together to form a point-to-point communication medium between the IP cores of a NoC, while an intra-switch link is defined to be a physical path connecting two ports inside a switch for sending the packets through \([\text{Raik07}]\). Assuming a single fault, an algorithm is discussed here to locate the faulty intra-switch link in a mesh-based network. Table 3.4 summarizes the fault model and reduces the 20 possible faulty intra-switch links into only 5 possible fault classes. This is to simplify the explanation of the algorithm.

![Figure 3.8: Input/Output ports of a switch with East→North intra-switch link.](image)

The flowchart of the algorithm is shown in Fig. 3.9 assuming only single fault in the network. The first fault class is tested by sending test packets through the horizontal links. If the test fails then it is known that there is a faulty \(E \rightarrow W_{i,j}\) link in this row. The test fails if the external output of the network at the west side doesn’t match the externally applied test input at the east side. To identify the column, XY-routing packets is applied sequentially across this row till a fault is detected. If there is no fault detected for the ‘n’ columns of the NoC, this implies that the last horizontal \(E \rightarrow W_{i,n}\) link in this raw is the faulty intra-switch link. However, if a fault is detected at column ‘k’, this implies that the link \(E \rightarrow W_{i,k-1}\) is faulty. If there was no horizontal faulty intra-switch links in the row, XY-routing test is also
Table 3.4: Classes of faulty intra-switch links in a switch.

<table>
<thead>
<tr>
<th>Fault</th>
<th>Equivalent faults</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E \rightarrow W$</td>
<td>$W \rightarrow E$, $S \rightarrow N$, $N \rightarrow S$</td>
<td>VH</td>
<td>Straight paths</td>
</tr>
<tr>
<td>$E \rightarrow N$</td>
<td>$W \rightarrow N$, $E \rightarrow S$, $W \rightarrow S$</td>
<td>XY</td>
<td>XY-routing</td>
</tr>
<tr>
<td>$N \rightarrow E$</td>
<td>$S \rightarrow E$, $N \rightarrow W$, $S \rightarrow W$</td>
<td>YX</td>
<td>YX-routing</td>
</tr>
<tr>
<td>$N \rightarrow R$</td>
<td>$E \rightarrow R$, $S \rightarrow R$, $W \rightarrow R$</td>
<td>RI</td>
<td>Resource inputs</td>
</tr>
<tr>
<td>$R \rightarrow S$</td>
<td>$R \rightarrow W$, $R \rightarrow N$, $R \rightarrow E$</td>
<td>RO</td>
<td>Resource outputs</td>
</tr>
</tbody>
</table>

Figure 3.9: Flowchart of the algorithm to diagnose a single faulty intra-switch link.
applied to detect faults in the $E \rightarrow N_{i,j}$ links. Similarly for $N \rightarrow E_{i,j}$ links by YX-routing. Finally the resource input and output intra-switch links are tested. This is done by writing a test value to each resource from a port and then reading it from all four output ports. Any mismatch at the outputs will locate the faulty link from the resource to an output port. However, if values of all the outputs are different from the input, this identifies that there is a fault in the intra-switch link connecting the input port and the resource. All the four possible input ports are applied sequentially in this test. This algorithm is applied recursively for all the rows in the network.

For example, suppose a $3 \times 3$ mesh-based network. Referring to Fig. 3.10(a), this network has a single faulty intra-switch link at $E \rightarrow W_{1,2}$. The diagnose algorithm starts at the first row by sending test packets horizontally (VH-Test) from the east to the west as depicted in the flow chart. Fig. 3.10(b) shows that the test fails, indicating that there is a faulty $E \rightarrow W_{1,?}$ intra-switch link. In order to know the column number, XY-Test is applied recursively for each column. Fig. 3.10(c) shows that the XY-Test fails at the third turn (k=3). As illustrated in the flowchart, this implies that the faulty link is at column 2 (k-1). Therefore, the faulty intra-switch link is located successfully at $E \rightarrow W_{1,2}$ by the above discussed diagnose algorithm.

![Network Diagram](image)

Figure 3.10: (a) $3 \times 3$ mesh-based network with faulty intra-switch link, (b) VH-Test fails, (c) XY-Test fails at the third turn.
Chapter 4

Overall Test Strategy

In this chapter, a composite test strategy is developed which utilizes the testing and diagnosis schemes that were discussed in the previous chapters. The key point from this test sequence is to combine different paradigms while taking into consideration the interdependency between them. This enables us to exactly identify and locate the faulty component in a NoC.

The test sequence starts by testing the inter-switch wire segments of the NoC. This is to ensure that the communication medium of the network is functional for later use of these interconnects as a TAM. Worst case fault model is being addressed in this method to ensure a high quality at-speed testing and relatively very low residual faults after testing. As shown in Fig. 4.1(a), the dedicated BIST structure for each inter-switch link is chosen for this test step. This is because other testing schemes assume that the switches are functional although they are not tested yet. This testing structure allows also fault diagnosis due to having a dedicated BIST for each link.

Since the FIFO buffers are small and distributed all over the chip. In the second step shown in Fig. 4.1(b), a concurrent test for all the FIFO buffers is applied. The shared BIST controller sends the functional test pattern discussed in section 3.2 through a dedicated bus medium to all the FIFO buffers. The time needed for this test is linearly proportional to the size of the buffers. Moreover, this test pattern supports fault detection as well as identification. This enables us to use the inter-switch wire segments and the FIFO buffers for testing the remaining components.

The third step shown in Fig. 4.1(c) is to test the RLBs. Multicast method as an external test is used to ensure exhaustive reuse of the internal channels to reduce the test time. In the test mode, the packets that are broadcasted through the network carry information about the test vector as well as the test response. Assuming that all switches are similar, thus, the test responses should also be similar. Each switch is designed to report back to the test
source a uniquely defined positive or negative acknowledgement based on the comparison between the incoming and the calculated test response. Since the broadcast tree is statically configured in the test mode. Therefore, the test source decides up on the faulty RLB based on the feedback values of the neighboring switches. For example, if switch-2a and switch-2b have negative and positive feedbacks respectively. Therefore, the RLB in switch-2a is faulty and vice versa. However, if both switches have negative feedback, this implies that switch-1 has the faulty RLB. If the test source perceive any feedback value that is different from the uniquely defined ones or if it does not receive an acknowledgement from a certain switch, therefore, most probably there is a fault either in the comparison circuitry or in the switch MUXs and intra-switch links. Consequently, the next step is applied to test and diagnose faulty MUXs and intra-switch links.

Finally, the last step shown in Fig. 4.1(d) is to test and diagnose faulty MUXs and intra-switch links in the network. The reuse of the interconnect infrastructure as a TAM reduces the pin and area overhead drastically. In this step, the three testing configurations and the diagnose algorithm discussed in section 3.4 are applied. This step is limited to mesh-based networks only.

Figure 4.1: Composite test strategy for testing NoCs: (a) Step 1: Testing inter-switch links, (b) Step 2: Testing FIFO buffers, (c) Step 3: Testing RLBs, (d) Step 4: Testing MUXs and intra-switch links.
Chapter 5

Conclusion

Network-on-chips became the key technology for integrating many-core processors on a single chip. This interconnects infrastructure enables a high performance and a high energy efficient system design. However, with the aggressive technology miniaturization and the demand of integrating billions of transistors on chip, defect densities are significantly increasing. For high revenue, the manufacturer should fabricate high volumes within a short time and with low cost. Therefore, the dissertation is focused on reducing the test cost since it has a substantial effect on the chip production.

This paper discusses several alternatives for testing and diagnosing all the basic building blocks of arbitrary NoC topologies. The key idea of reducing the test area overhead is to reuse the already existing pins and interconnects for testing the chip. On the other hand, the idea for reducing the testing time is to employ inherent parallelism in testing. By choosing the appropriate fault model and testing scheme, the probability of the residual faults after test is significantly reduced. The test starts by assuming the maximum aggressor fault model for testing inter-switch wire segments. Thereafter, those tested segments are used to transport test data towards the switches under test in a recursive manner. For each component in the switch, dedicated fault models are assumed and the appropriate test pattern is applied for fault detection and diagnosis.

Finally, NoC testing is a challenge for nanoscale technologies to increase their reliability by having a lower probability of failure. This enables in the future the use of nanoelectronic systems in critical applications such as medical equipments, transportation and nuclear technologies [Grec08].
Bibliography


