Reuse-Based Test Access and Integrated Test Scheduling for Network-on-Chip*

Chunsheng Liu† Zach Link† D.K. Pradhan §
† Computer and Electronic Engineering
University of Nebraska-Lincoln
Omaha, NE 68182, USA
{chunshengliu}@unlnotes.unl.edu

§ Computer Science
University of Bristol
Bristol BS8 1UB UK
pradhan@cs.bris.ac.uk

Abstract
In this paper, we propose a new method for test access and test scheduling in NoC-based system. It relies on a progressive reuse of the network resources for transporting test data to routers. We present possible solutions to the implementation of this scheme. We also show how the router testing can be scheduled concurrently with core testing to reduce test application time. Experimental results for the ITC’02 SoC benchmarks show that the proposed method can lead to substantial reduction on test application time compared to previous work based on the use of serial boundary scan. The method can also help to reduce hardware overhead.

1 Introduction
As an alternative to traditional System-on-chip (SoC) design, Network-on-chip (NoC) has been proposed [7, 17] as a new paradigm. It relies on an on-chip network to provide high performance interconnection to embedded cores. It has been shown that NoC is superior to other traditional architectures, especially for systems containing a large number of cores [18]. Hence NoC can potentially become the preferred interconnection scheme for the next generation SoCs. In this paper, we use the term ”NoC” to denote an on-chip interconnection network consisting of routers and channels, and the term ”NoC-based system” to denote the entire SoC system consisting of NoC and the functional embedded cores.

Testing of the embedded cores in NoC-based systems poses considerable challenges. In traditional SoC, test data are transported through a dedicated test access mechanism (TAM). In an NoC-based system, however, the implementation of network components, such as routers, channels, etc., has already consumed a significant amount of area. Therefore, new approaches for testing NoC-based systems rely on the reuse of the existing on-chip communication infrastructure without introducing new hardware overhead. It has been shown that the reuse methodology can significantly reduce both hardware overhead and test time [6, 12].

On the other hand, the network itself, i.e. routers, channels etc. have to be tested prior to the testing of embedded cores. Prior work [1, 2] treats routers as individual circuits and tests them through a dedicated TAM, e.g. serial boundary scan. The dedicated TAM may require additional hardware overhead and serial scan may lead to significant test time. As the complexity of router increases, efficient test solution is needed to deliver easy test access and reduced test time, without burdening the system with additional overhead.

In this paper, we present a new approach for test access and test scheduling of routers and functional cores in an NoC-based system. We present a progressive method to gain test access to the routers and schedule router testing by reusing the on-chip network. Test data are transported using network I/O pins and channels without the need of a dedicated TAM and additional test control pins on the chip. Test time can also be reduced. We also present possible hardware structure and functionalities required to facilitate the testing of the routers. We show that router testing can be integrated with embedded core testing, which also reuses the network for test data transportation, to reduce test application time. We present a scheduling algorithm that can concurrently schedule the tests of cores and routers. Experimental results on NoC benchmarks show that the proposed method can significantly reduce the test time compared to the prior work using serial boundary scan [1]. In addition, the integrated scheduling method further reduces test time. The NoC platform in this paper is based on the prior work [12] but the contribution of this paper is unique since the test of routers is not considered in [12].

Note that in this paper we focus on the test access and test scheduling of routers and cores at NoC level. Therefore, topics such as fault coverage, test pattern generation and fault diagnosis etc. of routers are beyond the scope of this paper and will not be discussed. These topics are recently covered by [2]. For the sake of conciseness, we also exclude the various constraints such as precedence and power constraints from consideration, which are addressed in [6, 12].

The rest of the paper is organized as follows. Section 2 lists some related prior work. Section 3 reviews basics of NoC. In Section 4 we present a new test access strategy for router testing and corresponding test scheduling strategy. Section 5 introduces an integrated test scheduling. Experimental results are presented in Section 6.

2 Prior Work
Traditional SoC testing relies on the use of dedicated TAMS, such as Test Buses or TestRails to gain test access. The design and optimization of dedicated TAMS and the corresponding test scheduling have been the focus of numer-
Reusing the on-chip network as TAM in NoC does not require extra hardware, since the functional cores are already adapted to the network by network interfaces. Therefore, the reuse approach provides a cost-efficient solution to the NoC-based system testing. The reuse of on-chip network as TAM for NoC has been studied in [6, 12] with preemptive and non-preemptive scheduling, respectively. Industrial NoC-based systems have also been introduced through several implementations [4, 17].

The test of on-chip network components, or NoC itself, was discussed in [1, 2]. In [1], a three-step methodology for testing a large number of identical processors, routers and memory blocks was presented. However, the method is not directly applicable to heterogeneous systems. Moreover, the use of serial boundary scan for router testing requires dedicated on-chip hardware for test access and additional test control pins, which may lead to additional area cost and pin counts, increased routing difficulty and long test time. Most recently, a scalable test methodology is presented for router testing [2]. The test access to router is done via a single pin, which is similar to that in [1].

3 Networks-on-Chip Basics

NoCs typically rely on the message-passing communication model in normal operation. The embedded cores exchange data in the network by sending and receiving request and response messages. Messages can be split into smaller structures named packets, which can be individually routed. Similarly, in testing mode of an NoC system based on the network reuse approach, test data are organized into test packets, which are routed through the network. The size of packets depends on the specific routing mechanism, buffer size etc. In this paper, we assume our network is able to contain either test data for one test load or test response from one unload in a packet.

Current embedded cores also need to use test wrapper to adapt their test infrastructure to the target NoC. In this paper, we assume the wrapper design techniques proposed in [10] is used. Besides the traditional test wrapper, additional hardware is required in NoC to provide interface between cores and NoC for protocols and other network functionalities, e.g. packing and unpacking of packets. Note that this hardware is implemented for normal operation, but it will be reused for testing.

In this work, we base our analysis on an existing packet-switching network model, so-called SoCIN [18]. It is implemented in a 2-D mesh topology. Figure 1 shows the implementation of the system d695 from the ITC’02 SoC Test benchmarks suite [13] in this topology, where two shaded cores are being tested using the dedicated path approach [12]. (The details of router implementation are shown in [2, 6], but not discussed here). It shows that the on-chip switching network can be reused for core testing. The communication channels between two adjacent routers are defined to be 32-bit wide. Each router in the SoCIN network is implemented using 3,000 to 6,000 gates, depending on the bit width of the network channel and depth of the input buffers [18]. The network uses credit-based flow-control and XY routing. Switching is based on the wormhole approach, where a packet is broken up into multiple flits (flow control units). Unicast network is assumed in this work.

4 Test Access and Test Scheduling for Routers

In order to reuse the on-chip network to test functional cores, routers have to be tested first. In this section, we show that routers can also be tested by reusing the on-chip network, which delivers test parallelism without the need of dedicated TAMs. Note that since the buffers associated with each router are usually tested by separate memory testing approaches, they are not discussed here. Similar to the boundary scan approach in [1], we assume that all the routers are
we present a progressive method for router testing. In this
reported. Therefore, a careful test schedule is needed. Here
if a router fails, the routers on the routing path have to be first
are in functional mode.
will be discussed later. Note that the router under test is con-
and control can be reduced.
will be processed, either on-chip or transported off chip, as
are then processed, either on-chip or transported off chip, as
an input port, routed through a number of routers and chan-
channels, and delivered to the router under test. Test responses
are then processed, either on-chip or transported off chip, as
will be discussed later. Note that the router under test is con-
in normal operation mode and are used to transport test packets to the next group of routers that are accessible. We
in this way, we avoid using untested routers in the routing path. Hence if a router fails a test, it can be immediately identified.
Figure 3(a) shows such a scheme for testing the routers in benchmark d695, where routers with the same number are in the same group and can be tested simultaneously. Note
that routers are organized into several groups, and each group can be directly accessible from the previous group or input
ports. As shown in Figure 1, although the system contains only 10 cores, all 12 routers need to be tested. The network
in Figure 3(a) supports multicast traffic, where test data can be delivered from one source router to multiple target routers simultaneously. In Figure 3(b), however, only unicast traffic is supported by the network, hence one source router can only access one target router at a time. We note that the use of multicast or unicast is determined by the network designer. We assume unicast in this paper.
In the reuse scheme introduced above, the test responses
of each group of routers under test are generated simultaneously. They then need to be processed to determine the test
result. Several approaches can be used to handle the test re-
sponses.
First, the response packets can be routed to the output
ports and downloaded to ATE for off-chip analysis. A bypassing mechanism is required in this case, such that the re-
sponse packets can be sent back to the network when the cur-
rent router is in test mode. Moreover, the routing path may
include untested routers, which can contaminate the results
in the packet. Therefore, this method is only possible for a pass/fail test, but not for diagnosis.
Secondly, test responses can be compacted into a MISR
(Multiple Input Shift Register) for each router or a group of
routers, and the signature in MISR is analyzed at the end of
testing. The signature can be either compared on-chip with a reference signature, or downloaded for off-chip analysis.
This method does not involve the use of untested routers, but
it could lead to aliasing due to the compaction in the sig-
nature. A minimum amount of extra hardware is adequate
because signature is usually very short.

Figure 3. A progressive scheduling for router testing: (a)
multicast network, (b) unicast network.
Finally, we can use the method proposed in [1, 2], i.e., responses from a group of routers are compared to each other using an on-chip comparator. Although the implementation of comparators increases hardware overhead, it is preferred for fast and accurate testing. Similarly, the result of comparison can be downloaded using a minimum amount of extra hardware. Note that aliasing is also possible in this case.

In the test scheduling scheme shown in Figure 3, the on-chip implementations are assumed. I.e., either MISR or on-chip comparators are used to handle the test responses. We hold this assumption for the subsequent discussion. The on-chip response processing for router testing is depicted in Figure 4. Note that the specific techniques for MISR compaction and comparators, e.g. X-compatible compaction, probability of aliasing etc., are important for a practical solution but they are beyond the scope of this paper and are not included.

4.4 Test wrapper

As discussed earlier, test wrapper in NoC is similar to the IEEE Std. 1500 compliant wrapper used for traditional SoC testing [14]. We note that a router and its associated core may each have a separate test wrapper. However, in practice router is usually integrated with its associated core [1]. Hence an integrated wrapper that wraps both core and router is possible.

A simplified architecture of such a test wrapper in test mode is shown in Figure 5(a). It is very similar to the standard Std. 1500 compliant wrapper but it can be used for both router testing and core testing. In Figure 5(b), we show the configuration of router and core in mission mode (normal operation). Also shown in the figure is a packing/unpacking mechanism. It is required in mission mode, because the incoming data must be retrieved from the packets, and the outgoing data must be assembled into packets before being sent to the network. However, Figure 5(a) shows it can also be reused for test packets processing in test mode, such that when the router is in test mode, test vectors can still be accepted and test responses can be sent out.

5 Integrated Test Scheduling

Test scheduling for embedded cores in NoC-based system is presented in [6, 12]. Here we present a test scheduling algorithm that can integrate router testing with core testing. It is based on the use of the dedicated routing path approach proposed in [12], where a routing path and a pair of input/output ports are dedicated to the test of a core (or router) until it is finished. Note that various constraints such as precedence and power can be implemented on top of this algorithm as in [6, 12]. They are not included in this paper due to the lack of space.

It is intuitive that one can first test all the routers using the progressive method presented in Section 4.3, then start core testing. This ensures that the untested routers will not be used for core testing. (Note that as we have assumed in Section 4.3, test responses are processed on-chip.) However, the scheduling can be made more efficient by observing that cores and routers can be tested concurrently. In the unicast network shown in Figure 3(b), we notice that after three groups of router testing, the routers in the bottom row are all tested. Therefore, input 2 and output 2 can be used to test the three cores in the bottom row. Also note that routers in group 4 can be tested concurrently using input 1 and the routers in the row next to the bottom, without conflicting with the core testing.

This example shows that after some groups of routers are tested, there may exist a path from a free input port to an output port, such that the cores along this path can be tested concurrently with routers in the remaining groups. In Figure 6, we present a pseudo code that integrates the router testing with core testing using the dedicated routing path approach.

The algorithm is similar to the one proposed in [12], but integrates router testing together with core testing. It starts by creating an ordered core list (with decreasing order of test time) as well as an input/output (I/O) pairs list. The orders of I/O pairs in the list are permuted and every permutation is attempted. The algorithm maintains a time tag on every re-
6 Experimental Results

In this section, we present simulation results on some NoC benchmarks that are crafted based on ITC’02 SoC Test Benchmarks [13]. We carry out experiments using both boundary scan in [1] and the proposed reuse method. All experiments are performed on a Sun Blade2000 workstation with 1.2Ghz processor. The execution of the heuristics requires only a few seconds of CPU time even for the largest system. The run time scales well with larger number of I/O ports and we expect similar results for large industrial NoC systems.

We calculate the test application time of router testing for both boundary scan [1] and the proposed reuse approach. We show only router testing time for boundary scan instead of overall testing time, because it is difficult to directly apply the multiprocessor test methodology in [1] to the ITC’02 benchmarks. Since benchmarks for router are not available, we assume router testing time is the average of testing times of all the cores in the system.

In order to make a fair comparison, the use of extra hardware for testing should be limited. Note that the proposed reuse method needs a minimum amount of extra hardware for testing. We allow boundary scan to use more hardware by assuming that all routers can be tested simultaneously, i.e., using maximal group of routers, see Figure 2. This is the best time that boundary scan can achieve, and it needs a significant amount of hardware overhead and additional pins. In addition, we note that although a parallel boundary scan seems to be possible [14], which is many times faster, it requires exceedingly high routing overhead and pin counts for test access, hence it is indeed impractical. This extra overhead renders the comparison even more unfair. Therefore, we assume a serial boundary scan is used in the experiments.

For the sake of completeness, we also calculate the router testing time using the TestBus approach as proposed in [10]. We assume routers are evenly distributed on test buses, the number of test buses equals the number of I/O ports used in NoC, and each test bus is 32-bit wide, which translates into 64 to 128 bit of total TAM width. However, we note that this result is only used for the purpose of reference. It is hardly practical for NoC-based system, because implementing a dedicated TAM of up to 128 bit, on top of the on-chip network, can cause prohibitively high routing overhead and large pin count. This is also discussed in details in [6].

For each benchmark, we give results for different numbers of input/output ports, shown in Column 1. Column 2 gives the router testing time using the boundary scan scheme in [1]. Column 3 shows the router testing time using the reuse-based method proposed in Section 4. We also show in parenthesis the reduction of test time in percentage, compared to those in Column 2. Column 4 shows the router testing time using test buses [10], as discussed earlier. In Column 5, we show the test time when routers and cores are scheduled separately using the proposed reuse method. Column 6 presents the results when router testing is integrated with core testing, as proposed in Section 5. We also show the percentage of test time reduction compared to those in Column 5.

It can be seen that the reuse method leads to a significant reduction on router testing time compared to the boundary scan scheme. As the number of available I/O ports increases, the reduction increases accordingly, because the reuse method can increase data throughput by using more I/Os in parallel. The use of test buses can lead to similar or sometimes lower router testing time than the proposed reuse method. However, as discussed earlier, it requires high overhead. We also observe that by concurrently testing routers and cores, the integrated scheduling algorithm can reduce the overall test time in all the cases, up to 23%, compared to scheduling the router and core testing individually. This

---

**Procedure NoC\_integrated\_schedule**

1. Sort cores in decreasing order of test time;
2. Permute all possible orders of I/O pairs;
3. For every permutation
4. do router testing while no free routing path for core testing
5. While (router testing incomplete and at least one I/O pair used for testing routers) or router testing completed
6. While there are unscheduled cores
7. For each unscheduled core
8. Find a free I/O pair;
9. If no free I/O pair
10. Update current time, repeat from 6;
11. else
12. Check the corresponding routing path;
13. If path is blocked
14. all cores have been attempted
15. Update current time, repeat from 6;
16. else
17. try next core in the list;
18. else
19. assign core to the path, update time tags;
20. update router status, update time tags.

**Figure 6.** Pseudo code of the integrated scheduling algorithm.
corroborates the effectiveness of our integrated test scheduling algorithm. Note that the effectiveness is also related to specific cores and the assignments of cores to network and I/O ports, which can be seen from the variation of results on different benchmarks.

### Table 1. Test scheduling result for d695.

<table>
<thead>
<tr>
<th># of I/Os</th>
<th>Router testing time</th>
<th>Overall testing time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Boundary scan</td>
<td>Network reuse</td>
</tr>
<tr>
<td>2/2</td>
<td>65970</td>
<td>26376</td>
</tr>
<tr>
<td>3/3</td>
<td>65970</td>
<td>22608</td>
</tr>
<tr>
<td>4/4</td>
<td>65970</td>
<td>18840</td>
</tr>
</tbody>
</table>

### Table 2. Test scheduling result for p22810.

<table>
<thead>
<tr>
<th># of I/Os</th>
<th>Router testing time</th>
<th>Overall testing time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Boundary scan</td>
<td>Network reuse</td>
</tr>
<tr>
<td>2/2</td>
<td>306525</td>
<td>295992</td>
</tr>
<tr>
<td>3/3</td>
<td>306525</td>
<td>197328</td>
</tr>
<tr>
<td>4/4</td>
<td>306525</td>
<td>147996</td>
</tr>
</tbody>
</table>

### Table 3. Test scheduling result for p93791.

<table>
<thead>
<tr>
<th># of I/Os</th>
<th>Router testing time</th>
<th>Overall testing time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Boundary scan</td>
<td>Network reuse</td>
</tr>
<tr>
<td>2/2</td>
<td>1999300</td>
<td>699128</td>
</tr>
<tr>
<td>3/3</td>
<td>1999300</td>
<td>523436</td>
</tr>
<tr>
<td>4/4</td>
<td>1999300</td>
<td>436955</td>
</tr>
</tbody>
</table>

#### 7 Conclusions

We have proposed a new method for test access and test scheduling in NoC-based system. We have presented a progressive reuse of the network resources for test access of the routers and an integrated test scheduling algorithm. Experimental results for the ITC’02 SoC benchmarks show that the proposed method can lead to substantial reduction on test application time compared to the approach using serial boundary scan.

#### References


