DfT for the Reuse of Networks-on-Chip as Test Access Mechanism

Alexandre M. Amory[1] Frederico Ferlini[2] Marcelo Lubaszewski[1] Fernando Moraes[2]

[1] UFRGS Federal University
Instituto de Informática
Av. Bento Gonçalves, 9500
Porto Alegre, RS, Brazil
amamory@inf.ufrgs.br, luba@ece.ufrgs.br

[2] PUCRS Catholic University
Faculdade de Informática
Av. Ipiranga, 6681
Porto Alegre, RS, Brazil
moraes@inf.pucrs.br

Abstract

This paper presents new DfT modules required to use networks-on-chip as test access mechanism. We demonstrate that the proposed DfT modules can be also implemented on top of low cost networks-on-chip, i.e. networks without complex services. The DfT modules, which consist of test wrappers and test pin interfaces, are designed such that both the tester and CUTs transport test data unaware of the network. We analyse the DfT modules in terms of silicon area and test time, considering different network and test configurations.

1 Introduction

Modular testing enables the test of embedded cores in a System-on-Chip (SOC) [15, 8]. It requires test logic, i.e. Design-for-Test (DfT) hardware, to ease the test access. The main DfT modules of a core-based design approach are test wrappers, used to switch between functional and test modes, and Test Access Mechanisms (TAMs), used to transport the test data from/to the test pins to/from the Core-Under-Test (CUT).

Traditional global on-chip interconnects, like buses, are becoming the bottleneck in terms of bandwidth, signal integrity, and power dissipation of complex chip designs [9]. A new communication architecture called Network-on-Chip (NoC) [2, 3] was proposed to solve these issues. In a NoC-based chip, cores communicate with each other via a network which consists of *network interfaces* (NI), *routers*, and channels to connect the routers. Data is transported in packets, which contain network dependent information required to route the data to its destination. Since, a conventional TAM is more similar to buses than to NoCs, it is also subject to signal integrity and power dissipation problems. One way to solve this problem is to use the functional interconnect (the NoC) as TAM [4, 5, 6, 12, 13, 14].

This paper presents a DfT environment described in RTL VHDL that enables NoC reuse as TAM. The main contributions of this paper are threefold. First, a new module called ATE Interface adapts the test stream to the NoC format and vice versa. Previous papers assume the tester has direct access to the embedded NoC, which is not realistic. Second, strategies are presented to eliminate the jitter (delay variation) in networks without guaranteed services, enabling the use of the test approach in low cost NoCs. Finally, a simulation-based approach is used to determine the DfT logic with minimal silicon area.

This paper is organized as follows. Section 2 presents previous papers about NoC reuse as TAM. The sources of jitter in NoCs are identified in Section 3. Section 4 presents the overall DfT strategy, while Section 5 details the design of the DfT modules. Section 6 shows area and core test time results under different network configurations and test bandwidth requirements. Section 7 concludes the paper.

2 Prior Work

Although Cota et al. and Liu et al. had considered different issues like maximum power dissipation in test mode [6, 5], thermal constraints [14], BIST, precedence constraints [12], and different clocks [13], a DfT model for NoC reuse as TAM was missing. DfT design and modeling for NoC reuse as TAM started with Li et al. [11] and Amory et al. [1]. Li et al. [11] propose a multiple data flit format and a scan chain configuration method to maximize the network utilization. The method allows wrappers to dynamically change the TAM width. Similar approach was previously proposed for conventional TAMs [10]. It is well known that configurable wrappers require a more complex control logic and more silicon area than conventional static wrappers, however, this extra area is not demonstrated in the paper. The authors do not show how they deal with load fluctuation, jitter, and traffic deformation [18]. These effects are sources of gaps in the test data delivery inherent to NoCs. Moreover, the authors state that scan testing based on NoC reuse is different from the scan test in conventional TAMs. *This paper demonstrates the opposite, that the NoC can be totally abstracted from the external tester if an adequate DfT logic is used*.

Amory et al. [1] present a test wrapper design for cores. The cores are assumed to have standardized interface and the network is assumed to have guaranteed communication service to guarantee test data delivery. However, there are two main issues in a test approach relying on guaranteed services. First, not all applications need such services. Second, guaranteed services require more complex NoCs and more silicon area, which can only be justified for complex designs. Thus, *it was not clear if the method could also be used in low cost networks*.

Both Li et al. [11] and Amory et al. [1] *assume that the test equipment is directly connected to the NoC*, which is not realistic. To mention one reason, let us assume, for instance, that a NoC is connected to the embedded cores by OCP protocol [17] (or any other protocol). It is well known that testers do not 'speak' OCP, thus, some protocol conversion logic is mandatory to keep the test data flow between the tester and the CUTs.

At the best of our knowledge, this is the first paper to present a complete DfT strategy to reuse a given NoC as TAM; the need and the design of test pin interfaces is first addressed in this paper. Moreover, unlike [1] and [11], we demonstrate NoC reuse using low cost networks and that the proposed test approach is compatible with current test practices.

25th IEEE VLSI Test Symmposium (VTS'07)
0-7695-2812-0/07 $20.00 © 2007 IEEE

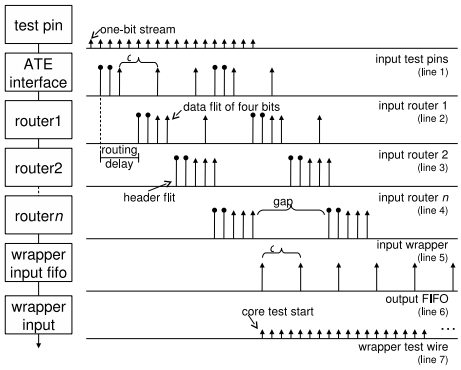

Figure 1: Example of load fluctuation in an input test path.

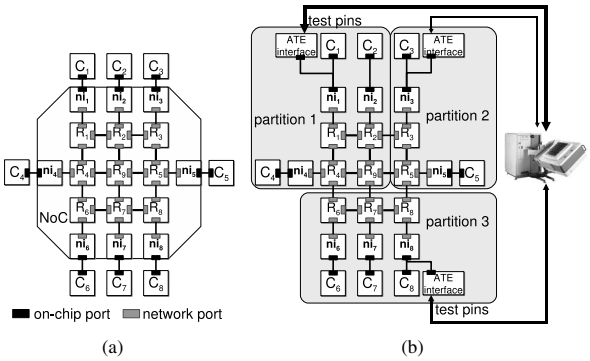

Figure 2: Test model based on NoC partition. (a) NoC-based SoC; (b) a valid partition.

3 Test Traffic and the Sources of Jitter in NoCs

Test traffic requires uncorrupted, lossless, in-order data delivery, zero-jitter, and scalable and constant bandwidth. The first two requirements are usually implemented in NoCs. The third one is easily accomplished by sending test data over a single path. Jitter is caused because NoCs have several shared resources (channels and routers). Shared resources require arbitration schemes, which cause jitter. Scalable and constant bandwidth is easily solved if the zero-jitter requirement is fulfilled; it only requires to send/receive data at regular time intervals.

In conclusion, zero-jitter requirement is the most critical to be met. This section identifies the sources of jitter in NoCs.

3.1 Shared Channels

Channels are shared in a network to save area for wiring. However, when two packets compete for the same channel, the arbitration logic grants access to one of them. The other stream is delayed, causing jitter and disrupting the test application of a core.

3.2 Shared Routers

The routing time of a router, i.e. the time it takes to find the output port, also causes jitter. There are two types of router schemes that are relevant for test: *routers with distributed and centralized routing* [7].

In the first case, the routing logic is typically located at the input ports, thus, it usually requires more silicon area to replicate the routing logic to every input port. However, it can route several independent packets (packets that do not require the same output channel) in parallel without any loss of performance. Therefore, the routing delay is deterministic and known a priori.

Centralized routing schemes save in silicon area but cause an indeterminate time to route a given packet. Usually, the packet that arrives first to the router is routed first, while the other routing requests wait. Therefore, the routing delay varies according to the router utilization.

Since it is not up to the test engineer to decide whether centralized or distributed routers are used in a design, the test strategy must support both types of routers.

3.3 Load Fluctuation

Figure 1 illustrates the test data flow from the input test pins to the input of the CUT. The input test pins receive continuous test data from the tester (line 1) and the CUTs expect the same traffic shape (last line). However, one can realize that the traffic shape at the input of router 1 (line 2) is different from the wrapper input (line 5). This deformation is called *load fluctuation* and it is caused by the routing delay, which in this example is four clock cycles. Thus, the data is 'condensed' every router, and, at the end of the flow the traffic has a bursty format (line 5). This traffic shape is not interesting for test because it creates gaps (line 5) in the data delivery. Then, a FIFO buffer is used to transform the previous bursty format to the periodic format (line 6). The periodic traffic shape is preferred because it is easier to transform it to the original traffic shape sent by the tester (lines 1 and 7).

4 Overall DfT Strategy

FIFO buffers (line 6 of Figure 1) are typically used to eliminate the jitter. However, the size of the buffer is proportional to the jitter bound [18]. Thus, a small jitter bound implies in a lower silicon area for DfT. The rest of this section shows the proposed approach to reduce the jitter bound and to minimize the silicon area for DfT.

4.1 Reducing the Jitter Bound Using NoC Partitioning

Let us initially introduce an *ATE interface* as an interface between the test pins and the NoC (line 2 of Figure 1). The proposed solution is to sub-divide the network into different logic partitions. Suppose, that a directed graph models the entire NoC-based chip design, such that routers, NIs, and cores are nodes and the channels are edges. A *NoC partition* is a sub-graph of the system graph such that *there is only one ATE interface, which represents the test source and sink in the partition*, and *all nodes of the partition can be reached from this source and sink*. Modules of a NoC partition are tested sequentially. However, the entire system can have multiple NoC partitions, supporting parallel testing. A module belongs to only one NoC partition and every module must be in one partition. Figure 2(a) presents an example system and Figure 2(b) shows its three test partitions.

A test scheduling tool, which is not the focus of this paper, creates the test partitions such that the chip test time is minimized. Once the number of partitions and the number of test pins used by each partition is determined, the methods described in this paper are used to optimize and generate the proposed DfT logic.

4.1.1 Shared Channels and Routing Logic

The proposed NoC partition solves the problem of shared channels because there is only one test source and sink (the ATE interface) per partition causing no competition for channels and packet collision. With the proposed partition method it is possible to give guarantees of jit-

ter even in networks without guaranteed services because the proposed approach avoids the main source of jitter, i.e. shared channels.

The NoC partition approach also helps to solve the shared router problem because not more than two flows compete for router services: the stimuli test flow and the responses test flow.

4.1.2 Load Fluctuation

While NoC partition is used to reduce the jitter bound, the jitter is actually eliminated with FIFO buffers at the end of the data flow. The goal of the FIFO buffer is to receive a variable input rate (line 5 of Figure 1) and to create at the output a constant rate (line 6 of Figure 1). The FIFO works as an elastic buffer in which the data is temporarily stored and then retransmitted at a constant rate.

4.2 Virtual TAM over NoCs

The scheduling tool determines how many test wires each partition must use. Equation 1 is used to translate the number of test wires into bandwidth used by the network.

$$d = \left\lfloor \frac{w}{tw} \right\rfloor \quad (1)$$

where w is the network data width, tw is the required number of test wires, and d is the interval (in clock cycles) between successive data pulses. The example in Figure 1 requires one test wire in a network of four data width, thus, the data interval (see d in Figure 1) is four. Then, the DfT module is configured to send data (ATE interface in line 2) and consume data (FIFO in line 6) every four clock cycles. The data interval d represents and satisfies the constant bandwidth test traffic requirement.

This proposed approach creates the abstraction to the test equipment that the test data is transported like in dedicated TAMs. One can see that the test traffic shape of the first and the last lines of Figure 1 are identical, and it is also identical to the traffic shape of dedicated TAMs. Thus, the proposed approach is compatible with current TAMs. In addition, the proposed method is scalable in width like dedicated TAMs. Of course, the number of test wires tw is never bigger than the network data width w.

4.3 Minimizing Silicon Area of the DfT Logic

The optimization procedure finds the minimal FIFO depth f and packet size p such that tw test wires are sustained, reducing the DfT silicon area. The DfT modules, ATE interface and wrappers, can be configured in terms of tw, f, and p. The following simulation-based procedure is used to minimize silicon area.

Algorithm 1 [Minimize Silicon Area(tw)]

```
//   search the FIFO depth
01.  gap = true; f = 0; p = ∞;
02.  while( gap = true)
03.     simul_env(f, p, tw);
04.     if gap_found then f = f + 1; else gap = false;
//   search the packet length
05.  p = 1; gap = true;
06.  while( gap = true)
07.     simul_env(f, p, tw);
08.     if gap_found then p = p + 1; else gap = false;
```

For a given partition with tw test wires and a given environment, like the one illustrated in Figure 3, the algorithm finds the minimal f and p. The function *simul_env(f, p, tw)* runs a cycle-accurate simulation model of a partition, where this model can be configured in terms of f, p, and tw. The model checks, during simulation, if tw test wires is supported without gaps. The algorithm initially sets the packet length to the maximal size supported by the network (line 1). Next, the simulation model runs to test for gaps. In case gaps are found, the FIFO depth f is incremented (line 4). Once the minimal f is found, the next step iteratively finds the packet length in a linear search.

This algorithm relies on a cycle-accurate model for sake of accuracy. On the other hand, it is executed once per partition to reduce the CPU time because every module within a partition uses the same p and f configuration.

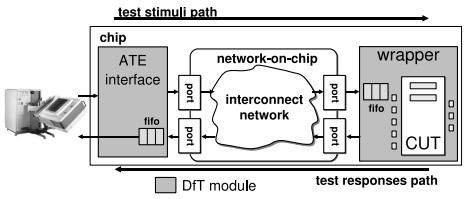

Figure 3: The test architecture of a partition.

5 Design of the DfT Modules

This section presents the DfT modules that give support to the proposed test model. The DfT modules comprise ATE interfaces and test wrappers. Figure 3 represents a NoC partition, where the DfT modules are connected by standardized protocol ports. The proposed modules use the previously mentioned FIFOs at the input of the wrapper and at the input (the network point of view) of the ATE interface.

5.1 ATE Interface

An *ATE interface* is the DfT module that connects the test pins to the NoC. In realistic designs, NoCs use standardized on-chip protocols, like OCP [17], which is different from the test protocol used by ATEs. Thus, some protocol conversion logic is required.

The main tasks of an ATE interface are to do *width conversion* between the number of test pins and the network data width, *protocol conversion* to communicate the ATE with the NoC, and *traffic shaping*, i.e. to correct the traffic deformation caused by load fluctuation.

The ATE interface illustrated in Figure 4 consists of two main blocks. The first one, called *ATE interface initiator*, is responsible for receiving the information from the input test pins and for inserting the packet header (CUT address in the network). The second one, called *ATE interface target*, receives the test responses from the CUT, removes the packet header information, and sends the actual test responses to the output test pins. The *width conversion*, from tw to w wires, is implemented by means of a shift-register. The *protocol conversion* must execute the NoC protocol and include packet header information in the test stream, like packet destination and number of words in the packet. Both NoC protocol and packet format are network dependent, thus, this part of the ATE interface also depends on the target NoC. The *traffic shaping* is carried out by counters that determine d (Equation 1) and by the FIFO used to eliminate the jitter.

5.2 Test Wrapper

Test wrappers are required to all modules of the chip, including cores, routers, and NIs (it may be convenient to combine a router with its NI to reduce the total number of wrappers). The basic principle of conventional test wrappers keeps the same, i.e. to receive stimuli, shift them to the wrapper cells and internal scan chains, apply the stimuli to the core, capture the responses, and send them to the tester. The main difference is that the network transports the test data. It implies

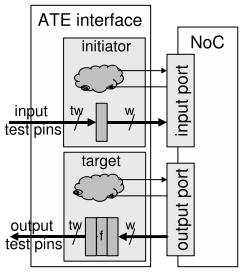

Figure 4: Simplified block diagram of an ATE interface.

that the wrapper has the same three tasks of the ATE interface (width conversion, implement the on-chip protocol, and traffic shaping) plus the tasks of a conventional wrapper.

The basic and common approach to design wrappers for NoC reuse is to define an input port, where test data comes in, and an output port, where the test data goes out the module. The words received from the source (ATE interface initiator) in the input port are shifted to the remaining core terminals and scan chains. After loading the complete stimuli, the test is applied and the responses are captured in the output wrapper cells to be shifted out to the output port. The output port sends the response to the sink (ATE interface target). The wrapper must know the network address of the sink to build a correct test response packet.

The only difference between the wrapper for cores and the wrapper for routers is that they use different protocols, which require slightly different wrapper control logic to execute the protocol. For instance, cores use OCP on-chip protocol and routers use credit-based router protocol (Figure 2(a) has different ports for cores and routers).

The proposed approach can load test wires faster than the conventional wrapper [8] because the last word received from the network, i.e. the word before applying the test pattern, does not need to be serialized as the previous words. Thus, the time to shift a test pattern can be shorter than the actual scan length. Equation 2 determines the scan time.

$$st_i = \left(\left\lceil \frac{sl_i}{d} \right\rceil - 1 \right) \times d + 1 \qquad (2)$$

where st_i, sl_i, and d denote the scan-in time, the scan-in length, and the time interval that the data is sent/read to/from the network, respectively. Similar equation also applies for the scan-out time (st_o).

The expression $\left\lceil \frac{sl_i}{d} \right\rceil$ (recall that $d = \lfloor \frac{w}{tw} \rfloor$) defines how many words of w bits are required to sustain tw test wires of length sl_i. For instance, if n words are required to transport a test pattern, $n-1$ of them take d clock cycles to be shifted while the last word is loaded in parallel (this parallel load is represented by the $+1$ at the end of the equation).

Finally, assuming a wrapper optimization algorithm is used to minimize the wrapper scan length sl_i and the Equation 2 is used to define the scan time st_i, then, the core test time T for a wrapped core is defined as

$$T = \{1 + max(st_i, st_o)\} \times pat + min(st_i, st_o) \qquad (3)$$

where *pat* denotes the number of test patterns, and st_i and st_o denote the scan-in and scan-out time for the wrapped core, respectively.

Compared to [1], this paper proposes to add a FIFO in the wrapper input port to remove the jitter. The FIFO and the partition method give guarantees even for NoCs without guaranteed services. As a consequence, even low-cost NoCs can use the proposed test method.

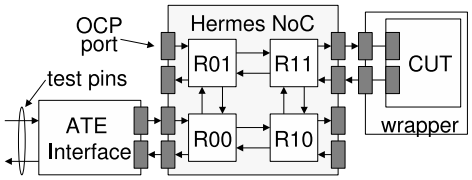

Figure 5: Experimental setup.

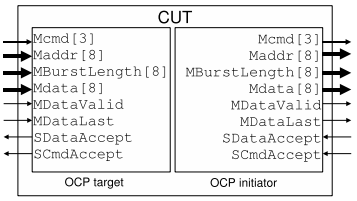

Figure 6: CUT with two OCP ports with 8-bit data width.

6 Experimental Results

The Hermes network is used as our case study [16]. Hermes is a parameterizable infrastructure used to implement low area overhead wormhole packet switching NoCs with 2D mesh topology. The router has centralized switching control logic and five bi-directional ports. Each input port has a FIFO for temporary flit storage. The network data width can be configured. It uses the XY routing algorithm.

Figure 5 illustrates the experimental setup. The NoC has been configured as 2x2 mesh network, but the number of routers of the network does not influence the results. The router FIFOs have 16 words and the NoC is OCP [17] compliant. This environment is modeled in RTL VHDL in a cycle-accurate HDL simulator.

We evaluate the proposed DfT changing network data width w and number of test wires tw. We show experiments with w equal to 8, 16, 32, and 64 bits. Thus, there are four instances of the system illustrated in Figure 5, each one with different network data width. The example CUT is a multiplication module with two OCP ports. Figure 6 shows the CUT with OCP interfaces configured with $w = 8$. The IO terminal count for each CUT instance is determined by $3 \times w + 7$. This core is connected to the NoC via the OCP ports. The CUT has no scan chains and $pat = 10$.

This case study is original because, unlike previous papers [1] and [11], it realistically assumes that the tester communicates with the NoC via test pins with protocol conversion logic.

The first column of Table 1 shows the network data width. The next three columns show the number of test wires the DfT logic must give support, the minimum packet length, and the minimal FIFO depth. We show these four data because they are the main variables that affect the design, the silicon area of the DfT, and the test time.

The minimal FIFO depth and packet length required to sustain the test wires are determined by the procedure described in Algorithm 1. The number of FIFO words and the packet length increase because, with more test wires, more bandwidth from the network is required.

All modules have been synthesized to 0.35μ technology library. The silicon area of the NoCs (without DfT) with 8, 16, 32, and 64 bits are, respectively, 23804, 43365, 78475, and 148454 equivalent gates. These sizes are used for comparison with the size of the DfT logic.

The following columns of the table are related to *silicon area to implement the DfT modules*. They show the area for the ATE interface

network data width	test wire	packet length	FIFO words	area (eq. gates)				test time				
				ATE interface (initiator)	ATE interface (target)	wrapper	% (prop/conv)	scan length	scan time (clock cycles)	conv. time (clock cycles)	prop. time (clock cycles)	% (prop/conv)
8	1	2	0	713	271	1023	8.4	31	25	351	285	-23.2
	2	5	2	724	538	1290	10.7	16	13	186	153	-21.6
	4	14	5	799	836	1566	13.4	8	7	98	87	-12.6
16	1	1	0	968	438	1558	6.8	55	49	615	549	-12.0
	2	2	1	974	584	1696	7.5	28	25	318	285	-11.6
	3	4	2	979	926	2043	9.1	19	16	219	186	-17.7
	4	5	2	976	926	2033	9.1	14	13	164	153	-7.2
	5	7	3	977	1080	2158	9.7	11	10	131	120	-9.2
	8	14	5	982	1440	2512	11.4	7	7	87	87	0.0
32	1	1	0	1476	767	2593	6.2	103	97	1143	1077	-6.1
	2	1	0	1475	752	2542	6.1	52	49	582	549	-6.0
	3	2	0	1472	708	2540	6.0	35	31	395	351	-12.5
	4	2	1	1487	1041	2846	6.8	26	25	296	285	-3.9
	5	3	1	1473	976	2858	6.8	21	19	241	219	-10.0
	6	4	2	1476	1669	3523	8.5	18	16	208	186	-11.8
	8	5	2	1493	1704	3513	8.6	13	13	153	153	0.0
	10	7	3	1476	1894	3726	9.0	11	10	131	120	-9.2
	16	14	5	1759	2559	4362	11.1	7	7	87	87	0.0
64	1	1	0	2535	1407	4615	5.8	199	193	2199	2133	-3.1
	2	1	0	2535	1392	4561	5.7	100	97	1110	1077	-3.1
	3	1	0	2524	1369	4548	5.7	67	64	747	714	-4.6
	4	1	0	2532	1376	4510	5.7	50	49	560	549	-2.0
	5	2	0	2510	1294	4509	5.6	40	37	450	417	-7.9
	6	2	0	2503	1291	4509	5.6	34	31	384	351	-9.4
	7	2	1	2522	2019	5148	6.5	29	28	329	318	-3.5
	8	2	1	2537	1952	5141	6.5	25	25	285	285	0.0
	9	3	1	2534	1920	5144	6.5	23	22	263	252	-4.4
	10	3	1	2505	1834	5153	6.4	20	19	230	219	-5.0
	12	4	2	2501	3195	6491	8.2	17	16	197	186	-5.9
	16	5	2	2540	3259	6481	8.3	13	13	153	153	0.0
	21	7	3	2501	3655	6893	8.8	10	10	120	120	0.0
	32	14	5	3061	4847	8092	10.8	7	7	87	87	0.0

Table 1: Results for the 8/16/32/64-bit system (prop = proposed, conv = conventional).

(both initiator and target) and the test wrapper for the CUT. The silicon area for the ATE interface initiator suffers a small variance due to different size of some internal counters, like the packet length counter. The ATE interface target is initially smaller than the initiator, but its area increases according to the number of FIFO words. The wrapper area is usually close to the area of an ATE interface since the protocol logic is very similar and the input part of the wrapper may also require a FIFO. The relative area overhead has a slight increase with more test wires. The highest area overhead is up to 13%. The systems with 16/32/64 bits have similar results to the system with 8 bits in terms of silicon area.

It can be observed that *the number of FIFO words and the packet length do not increase with the number of test wires, but it is proportional to the used network bandwidth*. Larger FIFOs are required only when the maximal bandwidth is used. For example, the NoC with 8 bits supports up to 4 test wires. However, the system with 64 bits supports up to 32 test wires. Although the maximal number of test wires of these systems are different, both of them require the same number of FIFO words and packet length.

The last five columns of the Table 1 show *test time results* for the CUT considering different number of test wires. They present the maximal scan length (scan in and scan out lengths are equal), the scan time to fill the wrapper cells, the core test time considering a conventional wrapper design, the core test time considering the proposed wrapper design, and finally the core test time reduction of the proposed method.

The wrapper optimization algorithm proposed in [1] is used to minimize the scan length. The scan time is calculated according to Equation 2. Using Equation 3 we realize that the proposed wrapper may test the CUT faster (up to 23%).

From Table 1, one can notice that the system with bigger data width requires cores with higher terminal count. This is observed by comparing the scan length of the 16-bit row and the 32-bit row for one test wire. It can be observed that the difference of core test time between the proposed and the conventional wrappers typically decreases as the number of test wires increases. This is explained because when the number of test wires increases, d decreases (see Equation 2). When d decreases, the advantage of loading the last word in parallel also decreases.

7 Conclusion

This paper presented DfT design required to support the reuse of a NoC as TAM. The proposed DfT approach consists of two test modules: a module called ATE interface used to connect the test pins to the NoC; and a test wrapper for every core, router, and NI.

The combination of the proposed network partition approach with the DfT logic gives the guarantees required by the test application. Therefore, the method can be applied even if the network does not implement guaranteed services. In addition, at the best of our knowledge, this is the first paper demonstrating NoC reuse working in a complete and realistic test scenario. Other relevant features of the proposed DfT method are: (*i*) the DfT logic hides the NoC from the tester, improving the compatibility with current testers and test tools; (*ii*) neither the cores nor the NoC are modified by the proposed approach; (*iii*) the DfT logic is modular, based on well-defined interfaces, scalable in test bandwidth, and configurable; (*iv*) it allows the trade-off between core test time and silicon area; (*v*) unlike previous papers, the proposed approach does not require complex NoCs, thus, it can be used also for designs with tight area constraints.

The results show that the proposed DfT requires more silicon area (up to 13%) compared to the conventional DfT for core-based design. However, our approach eliminates long wires which are subject to signal degradation, reliability issues, additional silicon area, more delay, more power consumption, and longer design time.

We hope that these results help to explain the advantage and drawbacks of NoC reuse compared to dedicated TAMs. In addition, we expect that the presented results of silicon area and core test time can be used in the future to generate more precise scheduling algorithms based on NoC reuse as TAM.

Acknowledgements

This work was partially supported by CNPq-PNM under scholarship grant 141993/2002-2. We acknowledge Erik Jan Marinissen and Kees Goossens for the discussion about core wrapper design.

References

[1] A. Amory et al. Wrapper Design for the Reuse of Networks-on-Chip as Test Access Mechanism. In *Proc. ETS*, pages 213–218, 2006.

[2] L. Benini and G. De Micheli. Networks on Chip: A New SoC Paradigm. *IEEE Computer*, 35(1):70–80, 2002.

[3] T. Bjerregaard and S. Mahadevan. A survey of research and practices on network-on-chip. *ACM Computing Surveys*, 38(1), 2006.

[4] E. Cota, L. Carro, and M. Lubaszewski. Reusing an On-Chip Network for the Test of Core-based Systems. *ACM TODAES*, 9(4):471–499, 2004.

[5] E. Cota et al. Power-aware NoC Reuse on the Testing of Core-based Systems. In *Proc. ITC*, pages 612–621, 2003.

[6] E. Cota et al. The Impact of NoC Reuse on the Testing of Core-based Systems. In *Proc. VTS*, pages 128–133, 2003.

[7] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks, An Engineering Approach*. Morgan Kaufmann Publishers, 2003.

[8] S. Goel and E. Marinissen. SOC Test Architecture Design for Efficient Utilization of Test Bandwidth. *ACM TODAES*, 8(4):399–429, Oct. 2003.

[9] ITRS. *International Technology Roadmap for Semiconductors*. Semiconductor Industry Association, 2005.

[10] S. Koranne. Design of Reconfigurable Access Wrappers for Embedded Core Based SoC Test. *IEEE Trans. VLSI Systems*, 11(5):955–960, 2003.

[11] M. Li et al. An Efficient Wrapper Scan Chain Configuration Method for Network-on-Chip Testing. In *Proc. ISVLSI*, 2006.

[12] C. Liu et al. Test Scheduling for Network-on-Chip with BIST and Precedence Constraints. In *Proc. ITC*, pages 1369–1378, 2004.

[13] C. Liu et al. Power-Aware Test Scheduling in Network-on-Chip Using Variable-Rate On-Chip Clocking. In *Proc. VTS*, pages 349–354, 2005.

[14] C. Liu and V. Iyengar. Test Scheduling with Thermal Optimization for Network-on-Chip Systems Using Variable-Rate On-Chip Clocking. In *Proc. DATE*, pages 1–6, 2006.

[15] E. Marinissen et al. A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores. In *Proc. ITC*, pages 284–293, 1998.

[16] F. Moraes et al. Hermes: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip. *Integration, the VLSI Journal*, pages 69–93, 2004.

[17] OCP International Partnership. *Open Core Protocol Specification. Release 2.0*, 2003.

[18] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proc. of the IEEE*, 83(10):1374–1396, 1995.