Evaluating the Scalability of Test Buses

Alexandre Amory, Matheus Moreira, Ney Calazans and Fernando Moraes
PPGCC, PUCRS, Porto Alegre, Brazil
{alexandre.amory, matheus.moreira, ney.calazans, fernando.moraes}@pucrs.br

Cristiano Lazzari
INESC-ID, Lisbon, Portugal
lazzari@inesc-id.pt

Marcelo S. Lubaszewski
PGMICRO, UFRGS, Porto Alegre, Brazil
luba@ece.ufrgs.br

***Abstract*—Intra-chip communication architectures evolved from buses to networks-on-chip, in order to provide design scalability and increased bandwidth. However, the predominant test architecture for SoCs is still based on buses. While this approach presents advantages, such as simple design and a mature set of automation tools, its scalability is questionable. This paper evaluates such aspect by synthesizing SoCs of different sizes (with more than 100 cores) to layout level and extracting accurate results in terms of wire length, capacitance, and delay. The results compare the wiring for test buses and for NoC links, indicating that these test buses have limited scalability (highly irregular wire lengths and long wires) and may not be suitable for testing future SoCs with hundreds of cores. Finally, we discuss advantages and drawbacks of some approaches proposed in the literature. This discussion might give directions towards new scalable SoC test architectural models.**

I. Introduction

SoCs emerged back in 2000, enabling the integration of 10s of cores into a single chip and requiring innovative SoC testing approaches. In this context, *Modular testing* has been proposed as a divide-and-conquer solution to test such complex SoCs [1]. Its conceptual model consists of: test wrappers to isolate the Core-Under-Test (CUT); test sources and sinks to send test stimuli and to receive test responses, respectively; and Test Access Mechanisms (TAMs) to transport test data from the test source to the CUT and from the CUT to the test sink.

Nowadays, SoC integration level continues to increase towards 100s of cores interconnected by Networks-on-Chip (NoCs), replacing buses which are not scalable enough for those designs. Some examples of highly integrated commercial ICs are: the Intel's 80-tile architecture [3] with die area of $275mm^2$, 100 million transistors, tile area of $3mm^2$, and width of $1.73mm$; and the Tilera's Tile64 [4] with 64 processor cores, die area has been estimated in $433.48mm^2$ ($20.82mm \times 20.82mm$), and width of $2.6025mm$ [5].

In the test domain, despite the issues related to buses, they are still the typical TAM approach. However, as more cores are integrated into a SoC, test buses might get longer, since they connect distant cores. For instance, let us assume the Tile64 design with 8×8 tiles of width $2.6025mm$. SoCs based on test buses might, in worst case, test two tiles in the corners of the chip in the same TAM. This would create a test bus of $2 \times 8 \times 2.6 = 41.6mm$. This bus is obviously too long, with huge capacitance and unacceptable delay.

The *motivation* of this paper is to present a quantitative evaluation of bus-based TAMs in terms of scalability. The studies related to the scalability of buses are limited to the chip's functional domain. This way, we expect to motivate more research on scalable test architectures. Bus-based TAMs are designed into MultiProcessor SoCs (MPSoCs) of different sizes (3×3, 5×5, and 9×9 tiles). These three MPSoCs are synthesized in $65nm$ technology and the results show that test buses might have long wires of few millimeters. Most of these long wires can be fixed by automatically inserting buffers. However, few long wires cannot be fixed with buffers, requiring manual insertion of flip-flops.

II. Prior Work

Basically there are two approaches to optimize the global wires of bus-based TAMs. The first approach is to *co-optimize test time and bus-based TAM wire length* [6]–[8]. These optimization approaches trade-off test time and TAM wire length, which means that wire length minimization is obtained at the expense of increased test time. Moreover, they require detailed and accurate layout information (floorplanning and position of core terminals) to perform the optimization. This means that the complete physical layout is necessary to finally generate the test architecture, delaying test planning until the last design phases. Furthermore, these previous papers rely on SoCs with less than 30 cores,

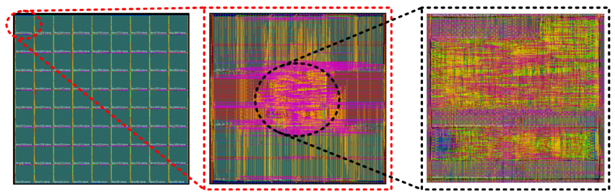


Fig. 1. Hierarchical layout of a 9×9 MPSoC. The left image represents the entire MPSoC. The center image illustrates the tile's layout. The right image details the tile's processor and router.

thus, they are not able to demonstrate scalability issues of bus-based TAMs.

The second approach is to *use existing functional interconnect to transport test data*. Using the existing functional interconnect for testing is a natural choice, in special for functional testing. The following works [9]–[11] use the existing on-chip bus to apply functional tests. On the other hand, if the chip already has a NoC connecting all cores, this NoC can be used to transport test data during the test application [12] and dedicated TAMs are dispensable. After a first proposal of using NoCs as TAMs [12], several improvements on test schedulers, test wrappers, and more general NoC TAM proposals [13]–[15] appeared. However, despite of all significant optimizations in terms of test time, the test approaches are far less standardized and mature than the test approaches based on dedicated test buses.

III. Physical Synthesis Setup

This Section presents the experimental setup (logical and physical syntheses and analysis setup) used to obtain the wire results in terms of length, capacitance, and delay.

In order to evaluate scalability, we adopted a MPSoC design named HeMPS [16] since its VHDL is available for download [1] and we can easily design systems of different sizes since it is parameterizable in terms of number of tiles. This MPSoC is homogeneous since it instantiates multiple identical tiles. Homogeneous MPSoCs are the most adopted design style, found in most academic MPSoC proposals and also in commercial chips such as Tilera's Tile64 [4]. A conventional bus-based TAM [1] is built to carry test data for each design. This bus is finally analyzed in terms of wire length, delay, and capacitance.

Figure 1 illustrates an example of MPSoC layout built for this experiment, which is detailed in this section. The layout has been divided into two steps presented next: the tile design and the SoC design.

[1]https://corfu.pucrs.br/redmine/projects/hemps

A. The Tile Design

A single hard macro block, named tile, has been designed to be the main building block of the SoC. The tile represents a HeMPS tile design with a 32-bit MIPS processor, network interface, DMA, and 16KB dual-port memory, and a Hermes NoC router (building a mesh-based network) configured with input buffers of size 16 and 32-bit wide NoC channels.

The tile's layout is represented by the image in the center of Figure 1. The HeMPS tile is synthesized with industry standard EDA tools, using ST-Microelectronics $65nm$ technology and standard cell library. 64 scan chains are added to the tile and the resulting HDL description is synthesized at the layout level. The result is a tile hard macro with area $967 \times 937 \mu m^2$, using 7 metal layers, where 5 layers are dedicated to the embedded memories. The tile interface terminals are placed according to the router ports orientation. West, East, North and South port terminals are respectively placed on the left, right, top and bottom borders of the tile. In this way, when placing the tiles according to the mesh topology, lower effort and resources are required to route the SoC. The local router port is already interconnected inside the tile, since the router and the IP are synthesized at the same hierarchical level.

The tile's layout in Figure 1 shows that the area is dominated by memory hard macros (dark green area in the background of the center image). The logic blocks are located at the center of the tile. The 6th and 7th metal layer are mostly dedicated to connect the router's terminals to the tile's terminals (see the wires over the memory macros). The right side of Figure 1 is a zoom in the tile's logic blocks. The tile is also synthesized with 64-bit scan chain. The scan-in terminals are located at the left side of the tile and the scan-out terminals are located at the right side.

B. The SoC Design

Once the tile design is finished, its hard macro can be imported to the SoC design such that it is instantiated several times, according to the SoC size. For instance, a 3×3 SoC has 9 tile instances. This hierarchical synthesis approach makes the SoC synthesis very efficient since it only routes global interconnect, i.e. functional and test wires connecting the tiles.

The first step to build the SoC is to decide its size (i.e. number of tiles). The ITC'02 SoC Test Benchmark is used as reference to build the evaluated SoCs. The

SoCs d695 (with 11 cores) and p22810 (with 29 cores) are used. Their cores were mapped to tiles organized on a 2D mesh topology with sizes 3×3 and 5×5, respectively, representing small- and medium-sized SoCs. With this setup we are basically assuming that: all ITC'02 SoC Test Benchmark cores have the area of our reference tile; there is a functional wrapper which adapts the benchmark's core interface to the NoC interface; no core requires more than 64 scan chains. If the core has less then 64 scan ports, the unused scan terminals are grounded.

Another issue is that the biggest ITC'02 SoC Test Benchmark has only 33 cores. Thus, a new SoC called 'big' has been created. This SoC has 117 cores (from the five biggest ITC'02 SoC Test Benchmarks named p22810, p34392, p93791, t512505, and a586710) which are mapped to a NoC with 9×9 routers. The cores of each of the three SoCs were randomly mapped to the NoC tiles such that no tile has more than two cores attached to it. Connecting the tile's functional terminals of a NoC-based design is straightforward since the NoC connects a tile to its neighbor tiles, guaranteeing short global functional wiring. On the other hand, connecting the tile's test terminals requires specialized tools, called test scheduler or test planner, to do it efficiently.

Several SoC test scheduling algorithms have been proposed and some of them were surveyed in [17]. However, despite their differences in terms of problem formulation, the basic goal of a test scheduler is to distribute the available number of SoC test pins among the cores such that the SoC test time is minimized. The output of the tool is a test partition where the cores grouped in the same TAM are tested sequentially and share the same subset of test pins. There might have multiple TAMs such that the test of each TAM occurs independently and in parallel. So, a test scheduling tool has been implemented based on the algorithm called TR-Architect [1]. It takes the test information (number of scan chains, scan chain length, number of I/O terminals) of each SoC benchmark (d695, p22810, and big) and the number of chip-level test pins to create a test architecture (set of TAMs) optimized for test time.

Figure 2 illustrates the resulting global level test interconnect for the d695 SoC assuming 16 test pins. The test scheduling tool partitioned the SoC into three TAMs, where the TAM1, TAM2, and TAM3 are, respectively, 2-bit, 6-bit, and 8-bit wide. The lines in the figure are a simplified representation the the TAM wiring layout, used only for illustrative purposes.

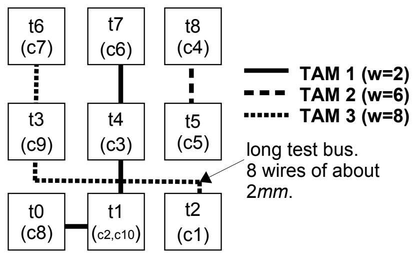


Fig. 2. Example d695 SoC with 16 test pins. It has three TAMs: with 2, 6, and 8 bits. Lines represent the TAM wiring layout. Tiles are numbered as t0 to t8 and cores are numbered as c1 to c10.

After the test scheduling, the generated test partition is translated to VHDL, connecting the tiles to the TAMs. The result is a top-level VHDL file instantiating the tile hard macros connected by NoC channels and bus-based TAMs. Next, the SoC is synthesized, generating the layout of the entire chip illustrated in the left side of Figure 1, assuming the 9×9 design. The tile hard macro are placed 200 μm from each other, given enough space to route the tile's interconnection. This tile space is referenced again latter in this paper. Finally, from the generated layout, top level SoC wire reports are generated. The resulting chip level wires are classified into NoC channel wires and TAM wires. Histograms are generated for each class of wires.

IV. Evaluating the Scalability of TAM Wires

The example presented in Figure 2 is a small SoC, but it shows that it might use long test wires. For instance, note that a long wire is required to connect c9 to c1. Assuming a regular floorplan with tiles of homogeneous area, and assuming the tile size of $967 \times 937\mu m^2$, *this 8-bit bus is approximately* $2mm$ *long. This section shows that larger SoCs, with more tiles, might have even longer wires*. The results are obtained using the setup presented in the previous Section and assuming the three SoCs with 64 test wires to connect each chip to the tester.

The first column of the charts in Fig. 3 shows frequency histograms of NoC channel wires. It can be seen that the NoC channel wire length varies from about 0 to 1 mm, due to global connections. However, it should be stressed that and most of the wires have about 200 μm of length, which is the distance among the tiles (Section III-B). The average NoC channel wire length for d695, p22810, 'big' is 340.98, 340.77, and 340.73 μm, respectively. It can also be observed that NoC channel wire length also remains fairly constant as the SoC size increases (compared the charts d695-channel with big-channel), demonstrating good scalability for NoC channel wires.

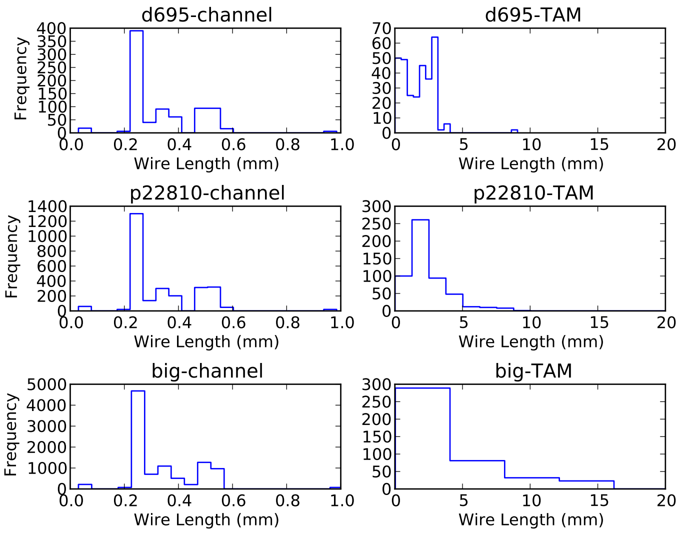
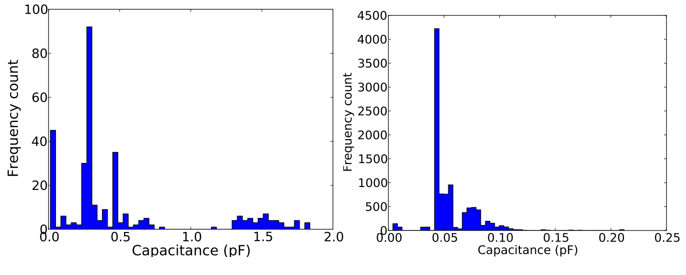


Fig. 3. Comparing TAM wire length (second column) with NoC channel wire length (first column) assuming SoCs of different sizes and 64 test pins.

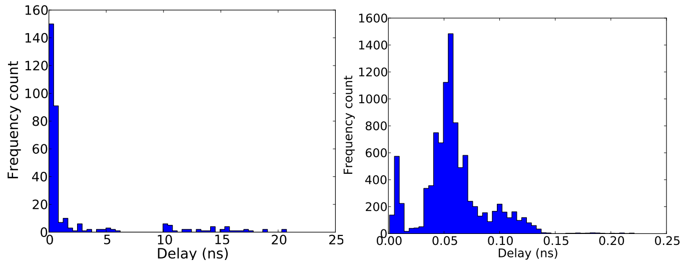
The second column of the charts in Fig. 3 shows frequency histograms of TAM wires. *It shows that the TAM wire maximum length increases with the SoC size (up to 4 mm in SoC d695, 9 mm in SoC p22810, and 16 mm in SoC 'big' which are, respectively, 4, 9, and 16 times longer than the longest NoC channel wire), demonstrating the lack of scalability of test buses.* The average wire length for d695, p22810, 'big' is 1774.99, 2440.70, and 2882.03 μm, respectively. Note that 2882.03 μm is more than 8 times longer than the average NoC channel wire length of about 340 μm. These wire length results are considerably worse compared to typical NoC wire length properties.

Classically, the length of a wire is directly related to its resistance. In this way, long wires will present higher resistance and, consequently, lower currents, which lead to increased propagation delays. In fact, in this technology, wires longer than 0.5 mm usually require buffers for signal regeneration. Another fact that contributes to the increasing propagation delay is the effect of the interconnect parasitic capacitances, inherent from metal layers, used for routing, separated by an oxide layer. The more congested is the routing of a design, the bigger will be the parasitic capacitances. Because big capacitances increase propagation delay, congested regions lead to wires with higher propagation delay.

In order to evaluate the parasitics capacitances of test buses, these values were extracted from the generated layout of the designed SoCs using Cadence Encounter. Fig. 4(a) shows the frequency count of the parasitic capacitances of test wires of SoC 'big', a worst-case for this experiment. As the chart shows, the majority of test wires present capacitances of 0.1 pF to 0.5 pF and in a worst case, these values increase to almost 2 pF. In addition, the results demonstrate that a considerably large portion of test wires present high parasitic capacitances, between 1.3 pF and 1.5 pF. In this technology, these values are considered high and represent roughly 1000 times the input capacitance of a minimum sized inverter, which is in the order of fF. This excessive parasitic capacitance, coupled to the elevated resistance caused by the length of wires, lead to increased delay figures. In addition, even more problematic, such capacitance values are over the maximum capacitance supported by the models of the employed library. In this way, in order to guarantee correct functionality of the design, such values need to be optimized, using techniques like buffer and flip-flop insertion.



(a) Capacitance of test wires (b) Capacitance of NoC wires



(c) Delay of test wires (d) Delay of NoC wires

Fig. 4. Capacitance and delay of both test and NoC wires.

After extracting the parasitics of the SoC, the resultant worst-case delays of their internal nets were annotated. Fig. 4(c) shows the frequency count of the propagation delay of test wires of the SoC 'big'. As the chart shows, the most frequent delay of the wires is of roughly 0.5 ns. However, for some wires, the annotated delay kept around 5 ns and for a relatively big portion of the wires, the delay varied from 10 ns to 22 ns. In this way the test of this SoC is limited to a frequency lower than 50 MHz.

Evidently, such delay and capacitance figures are not practical for high-end designs. In fact, in the case of such long wires with these parasitics, buffers are usually placed as repeaters for signal regeneration, in order to reduce propagation delay, and to reduce the capacitance of the resultant wires. However, albeit silicon area will typically not be affected, because the buffers can be placed in the gaps between tiles that were used for global routing, the use of these buffers will increase the SoC test power. Also, the bigger the length of the wire and the parasitic capacitances, the bigger is the number and size of the required buffers, leading to higher power values.

In order to estimate the number and size of buffers required for the test buses of the 'big' SoCs, Cadence Encounter was employed for automatically place buffers with different driving strengths from X2 to X284 on wires that did not respect maximum capacitance constraints. Also, it was defined a maximum delay of 2 ns, in order to allow test frequencies of 500 MHz. Recall that high-end chip designs use newer technologies which require test patterns for additional fault models rather than just the traditional stuck-at model. Thus, more test patterns are required for testing the SoC. In addition, as more cores are integrated into the SoC, the test data volume also increases. This way, increasing the test frequency is a trend [18] to reduce the SoC test time.

The experiment demonstrates that, from the total of 322 test wires, 266 did not require any buffer, 35 required one X2 buffer, 7 required one X4 buffer, 3 required 6 big buffers (with driving strengths bigger than X200) and 11 could not be fixed using only buffers. Another solution for the test buses would be to place flip-flops to allow more relaxed timing constraints. Even though, placing buffers and/or flip-flops has a considerable impact on the power and design complexity of the circuit. For instance, a single buffer with driving strength bigger than X200 has a static power of over 1.2 μW, while the estimated total static power of the memory blocks[2] of a single tile is of 80 μW, according to the memory's datasheet. This is a big overhead for a single buffer, 1.5%. Also, the need for flip-flops could only be observed after layout design, which indicates that this kind of optimization is only possible in late steps of the design flow. This is not desirable as it require a new logical and physical synthesis of the design, worsening time-to-market constraints.

Comparing these results with the ones obtained for NoC links' wires, it demonstrates that test buses are clearly less scalable. This scalability issue can be easily solved by automatic buffer insertion in small and medium sized SoCs, as the d695 and p22810 SoCs. However, in larger SoCs with hundreds of cores, the test wire length and capacitance might be too high for the used $65nm$ technology library, and flip-flops have to be inserted manually, delaying the design.

NoC wires, on the other hand, are scalable in terms of SoCs size. For instance, Fig. 4(b) and Fig. 4(d) show the frequency count of the parasitic capacitances and propagation delays of NoC links' wires of SoC 'big'. As the charts shows, the majority of the wires present parasitic capacitances lower than 0.15 pF and propagation delays lower than 0.15 ns. In fact, after the same analysis conducted to test buses, results suggested that none of the NoC links' wires require buffers or flip-flops.

V. Discussion and Future Perspective

A. About Silicon Area Overhead of Bus-Based TAMs

Most previous papers suggesting the use of NoC TAMs instead of dedicated bus-based TAMs argue that the main advantage of NoC TAMs is the silicon area saved by avoiding dedicated TAMs. Arguments tell, for instance, that the total TAM wire length could affect the chip level wiring congestion, potentially requiring more space between tiles, ultimately resulting in additional silicon area overhead.

Although dedicated bus-based TAMs may cause silicon area overhead, this was not observed in our experiments, where SoCs with and without test buses present roughly the same silicon area. One of the main reasons is that layouts were configured such that the space among tiles (200 μm) uses of all seven metal layers for routing NoC channels and TAM wires, which seems to be sufficient for routing all designs as we did, with no added area penalty. Even when placing buffers, the area should not be compromised, because they can be placed in the space among tiles. In conclusion, the silicon area overhead is the least relevant issue of test buses. The results of this paper suggest that scalability is much more relevant.

B. About Other Methods to Reduce TAM Wiring

Tools like [6], [7] are able to co-optimize both test time and wire length for bus-based test architectures, minimizing TAM wire length at the cost of additional test time. More recently, a similar layout-aware test co-optimization tool was proposed for testing 3D chips [8]. With the help of such test planning tools, it is still possible to use bus-based TAMs in medium sized

[2]A tile has a dual-port memory of 16Kbytes.

SoCs. However, although these can successfully reduce TAM wire length, we believe that they only reduce the scalability problem of bus-based TAMs, but are not able to eliminate it, and the problem will eventually appear again with bigger SoCs. Note that the biggest SoC evaluated in [6]–[8] has only 33 cores, while commercial SoCs already have hundreds of cores.

Perhaps, an alternative approach is to change the main optimization goal. Instead of co-optimize both test time and wire length (minimizing the wire length of all test buses) perhaps a better approach is just to avoid creating extremely long test buses. In addition, instead of using accurate physical synthesis information which could postpone the test optimization, a less accurate and simpler wire length estimation model, such as the Manhattan distance between CUTs, might be sufficient.

C. Perspectives for the Use of NoC TAMs

The results presented in this paper demonstrate a clear disadvantage between conventional bus-based TAMs and NoC channels in terms of scalability. These results indicate that the test domain might also need a more scalable communication architecture, such as NoCs. Using the existing NoC for test data transportation might be an efficient solution. However, much more research is required in the subject since NoC TAM proposals presented so far are tailored to a specific NoC design, typically with mesh topology and XY routing algorithm. More general NoC TAM approaches are required, which can be applied to any NoC design, independently of its topology, routing algorithm and so on.

D. Perspectives for the Use of Bus-Based TAMs

Despite the limited scalability of test buses, bus-based TAMs will probably continue to be the preferred approach for small to medium sized SoCs, due to their simpler design and since there are EDA tools supporting it. Moreover, for large sized SoCs (with a hundred cores or more), bus-based could be used TAMs locally, for example in a tile with multiple cores, and NoC TAM could be used globally, connecting the tiles.

VI. Conclusion

This paper sought to stimulate a discussion on the scalability of bus-based TAMs. The wire length of NoC channels and test buses were compared, leading to the conclusion that global test buses have highly irregular wire lengths with increasing parasitics as the SoC size increases, indicating lack of scalability of test buses. Results presented in this paper indicate that NoC TAMs inherit the scalability and modularity of NoCs, but require more research effort to become an industry reality.

References

[1] S. K. Goel and E. J. Marinissen, “SoC test architecture design for efficient utilization of test bandwidth,” *ACM TODAES*, vol. 8, no. 4, pp. 399–429, 2003.

[2] S. O. E. Lindholm, J. Nickolls and J. Montrym, “NVIDIA tesla: A unified graphics and computing architecture,” *Micro, IEEE*, vol. 28, no. 2, pp. 39–55, 2008.

[3] S. R. Vangal *et al.*, “An 80-tile sub-100-w teraFLOPS processor in 65-nm CMOS,” *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, 2008.

[4] B. E. S. Bell and J. Amann, “Tile64-processor: A 64-core soc with mesh interconnect,” in *Proc. ISSCC*, 2008, pp. 88–90.

[5] A. Lee and D. Killebrew, “L2 to off-chip memory interconnects for cmps,” Berkeley University, Tech. Rep., 2008. [Online]. Available: http://www.cs.berkeley.edu/~kubitron/courses/cs258-S08/projects/reports/project2_report.pdf

[6] S. K. Goel and E. J. Marinissen, “Layout-driven SoC test architecture design for test time and wire length minimization,” in *Proc. DATE*, 2003, pp. 738–743.

[7] E. Larsson and H. Fujiwara, “Test resource partitioning and optimization for SoC designs,” in *Proc. VTS*, 2003, pp. 319–324.

[8] L. Jiang, Q. Xu, K. Chakrabarty, and T. Mak, “Layout-driven test-architecture design and optimization for 3D SoCs under pre-bond test-pin-count constraint,” in *Proc. ICCAD*, 2009, pp. 191 –196.

[9] P. Harrod, “Testing reusable IP, a case study,” in *Proc. ITC*, 1999, pp. 493–498.

[10] J.-R. Huang, M. Iyer, and K.-T. Cheng, “A self-test methodology for IP cores in bus-based programmable SoCs,” in *Proc. VTS*, 2001, pp. 198 –203.

[11] A. M. Amory, L. A. Oliveira, and F. G. Moraes, “Software-based test for non-programmable cores in bus-based system-on-chip architectures,” in *Proc. IFIP Int. Conf. on VLSI*, 2003, pp. 174–179.

[12] E. Cota, L. Carro, and M. Lubaszewski, “Reusing an on-chip network for the test of core-based systems,” *ACM TODAES*, vol. 9, no. 4, pp. 471–499, 2004.

[13] E. Cota and C. Liu, “Constraint-Driven Test Scheduling for NoC-Based Systems,” *IEEE Transactions on CAD*, vol. 25, no. 11, pp. 2465–2478, Nov. 2006.

[14] J. M. Nolen and R. N. Mahapatra, “Time-division-multiplexed test delivery for NoC systems,” *IEEE Design & Test of Computers*, vol. 25, no. 1, pp. 44–51, 2008.

[15] A. M. Amory, C. Lazzari, M. S. Lubaszewski, and F. G. Moraes, “A new test scheduling algorithm based on networks-on-chip as test access mechanisms,” *Journal of Parallel and Distributed Computing*, vol. 71, no. 5, pp. 675 – 686, 2011.

[16] E. A. Carara, R. P. Oliveira, N. L. V. Calazans, and F. G. Moraes, “HeMPS - a framework for NoC-based MPSoC generation,” in *Proc. ISCAS*, 2009, pp. 1345–1348.

[17] Q. Xu and N. Nicolici, “Resource-Constrained System-on-a-Chip Test, a Survey,” *IEE Computers & Digital Techniques*, vol. 152, no. 1, pp. 67–81, 2005.

[18] ITRS, *Test and Test Equipment*, Semiconductor Industry Association, 2011.