Adapting an SoC to ATE Concurrent Test Capabilities

Rainer Dorsch[1], Ramón Huerta Rivera[1]
Hans-Joachim Wunderlich[1], Martin Fischer[2]

[1]Computer Architecture Lab, University of Stuttgart, Germany
[2]Agilent Technologies, Böblingen, Germany

Abstract*— Concurrent test features are available in the SoC testers to increase ATE throughput. To exploit these new features design modifications are necessary. In a case study, these modifications were applied to the open source LEON SoC platform containing an embedded 32 bit CPU, an AMBA bus, and several embedded cores. The concurrent test of LEON was performed on an SoC tester. The gain in test application time and area costs are quantified and obstacles in the design flow for concurrent test are discussed.

Keywords: ATE, Concurrent Test, SoC Test, Test Resource Partitioning

I. INTRODUCTION

A system-on-a-chip (SoC) typically contains various predesigned and prevalidated *embedded cores* like memory cores, processor cores, interface cores, mixed-signal, and analog cores. The system integrator embeds cores designed and validated once by a core designer in many systems. In order to facilitate the reuse of embedded cores, vendor independent documentation guidelines and widely accepted communication standards, like the AMBA, IBM core connect, MIPS PI (Peripheral Interface) or wishbone buses, have been developed [1–3]. Also the test of an embedded core is developed and validated once by the core designer and reused and integrated by system integrators in many SoCs. The system integrator is responsible for designing an appropriate *test access mechanism* (TAM) to transport the test data from the chip boundary to the embedded core under test [4–11].

Typically, each embedded core and each interface type in a hierarchical SoC design requires either a different test method or has at least a different test application time. Previously, tester limitations required a close to serial test of all embedded cores. The SoC testers on the market now may overcome these limits. An SoC tester has an additional port layer in its pin hierarchy [12]. Each port has the same setup possibilities as the device layer had in traditional testers.

An SoC tester may assign a port to each functional embedded core to test the cores concurrently, even if they require different test methods. For example, a properly configured SoC tester may activate and control a core containing memory BIST and at the same time run tests at a different frequency on a phase-locked-loop core located elsewhere on the chip. The only prerequisite is that the two cores do not use the same resource (ATE resource or IC resource, such as pin, wire, ...) in a conflicting way at the same time. *Concurrent test* of many embedded cores reduces test time for an SoC in many cases significantly [13,14].

In the next section, we review the tester requirements for concurrent test and introduce the design requirements of an SoC for concurrent test. To evaluate gains and costs of concurrent test and requirements for the EDA tools, we adapted the open source LEON SoC platform for a concurrent test on an Agilent 93000 SoC tester. In Section III the LEON SoC platform itself and the implemented modifications necessary to make it suitable for a concurrent test are described. Section IV discusses the used design flow which includes standard tools and user defined scripts. Also manual steps and problems in the design flow are discussed. The area costs, the impact on test application time and the test program development time of the concurrent test approach and a list of requirements to automate the design flow are given in Section V.

II. CONCURRENT TEST

Concurrent test reduces test application time by applying as many tests as possible at the same time. The test program development time is reduced, because the test pattern files for the cores tested concurrently need not to be merged by a test development engineer, but are loaded directly on the SoC tester. In order to perform concurrent test, both tester and design have to fullfil certain requirements.

A. Tester Requirements

In standard ATE, the common hierarchy reflected by the software in charge of building the device setup for the specific tester has three levels of abstraction as depicted in Figure 1. These levels were introduced to ease the programming of the tester. At the lowest level in the hierarchy we find the individual IC pin, for which I/O characteristics, voltage levels, timing shape of signals and test data may be configured. If we step one level up, we find the pin group which is used to manage the setup of pins which share common properties such as a memory data

0-7803-7542-4/02 $17.00

or address bus. Finally, at the topmost level in the hierarchy, we have the device under test (DUT) which corresponds to the whole IC.

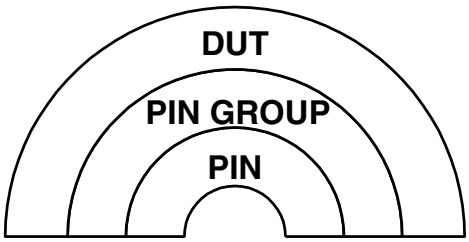

Fig. 1. Levels of hierarchy for standard ATE machines.

From the design point of view, SoCs introduce one additional level of hierarchy: the core level. This situation introduces a gap in the traditional ATE hierarchy which has to be filled in. For that reason, new generation testers include the so called test PORT in the ATE hierarchy, which is mapped with the embedded core in the chip design hierarchy, as shown in Figure 2.

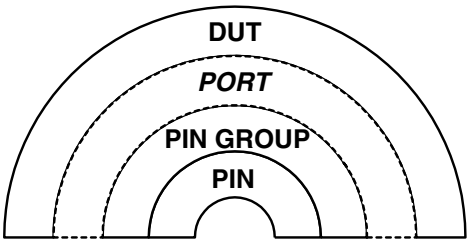

Fig. 2. Hierarchy present in new generation testers.

The test port allows the same setup possibilities as we had before for the DUT. This means that multiple port setups may be concurrently in the ATE as the DUT is now above in the hierarchy. The main implications regarding the ATE operation can be summarized as follows:

- **Multiple clock frequencies** are allowed to operate at the same time (each one assigned to a different port).
- **Multiple pattern files** are also possible to be applied concurrently to the same device (with no overlapping pin contained in them).

So the test port in the ATE will be highly correlated with the core in the SoC, providing the additional flexibility and performance required to deal with core-based designs as depicted in Figure 3.

B. Design Requirements

The *concurrent test approach* for core-based ICs tests as many cores as possible at system level as it is shown in Figure 4. The number of cores concurrently tested is often limited by the number of available chip pins or available tester channels. The concurrent test methodology is compatible with the design methodology of SoCs maintaining the hierarchy levels

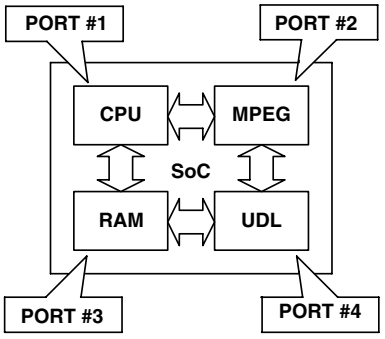

Fig. 3. Test scenario after the introduction of test ports.

contained in the IC, distinguishing the core level tests and translating them into system tests which will be core oriented.

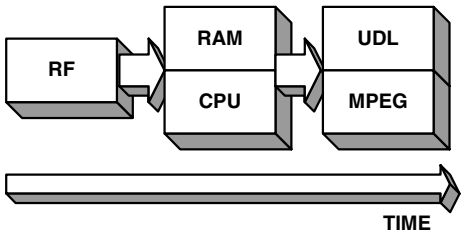

Fig. 4. Concurrent Test Approach

In order to test the blocks of an SoC concurrently, the design must be modified as follows:

- A test wrapper must be inserted which will isolate the core from its system environment during test [15–17].
- A test access mechanism (TAM) must be inserted, which will transport the test patterns from the chip pins to the cores and the test reponses from the cores to the chip pins [4–10].
- To the test of one or more cores a port must be assigned. If more than one core is assigned to a port, some control and clock signals may be shared.
- A multi-objective test scheduling must be done. The objectives of the scheduling are
  - to maximize the cores per test port in order to share control signals,
  - to maximize the throughput of the test access mechanism,
  - to minimize the test length, and
  - to keep the test power consumption below a certain limit.

The goal of this paper is to present a case study where all these tasks are performed manually in order to evaluate:

- Advantages in terms of shorter test application time.
- Advantages in shorter test program development time.
- Hardware costs for the required additional DfT.

- Additional features of EDA tools required to automate the DfT process and minimize the impact on the design.

III. LEON SoC PLATFORM

LEON is an open source SoC platform for embedded applications [18]. It was developed for space mission applications by the European Space Agency (ESA). We used it to develop a concurrent test for an SoC. The necessary modifications were performed manually in order to evaluate the methodology using a real design on a real tester.

A. Design

LEON contains an embedded 32 bit processor, an AMBA bus [2], and peripherals. The processor core is compatible with the Scalable Processor ARchitecture (SPARC) Version 8 [19] and includes an integer unit, a register file (embedded RAM) as well as data and instruction caches. The platform provides an interface for a floating point unit and a coprocessor. Moreover, several peripherals such as UARTs, timers, a memory controller, and an I/O port are included. The platform may be customized by connecting user defined cores to the AMBA bus. The block diagram of LEON containing the main parts of the design is shown in Figure 5.

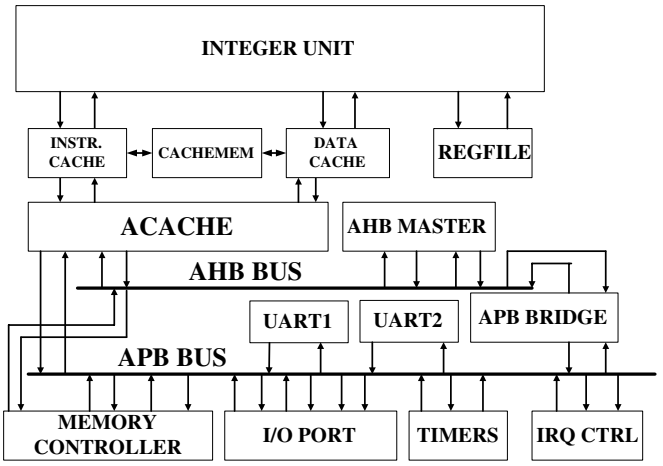

Fig. 5. LEON Block Diagram.

The interconnection scheme between the blocks is implemented by the Advanced Microcontroller Bus Architecture (AMBA). It consists of two on-chip buses: the Advanced High-performance Bus (AHB) for high data volume transactions and the Advanced Peripheral Bus (APB) for lower power consumption peripherals with low data transfer rates.

B. Block Test

The LEON platform comes without a detailed test concept. The starting point for test development was a set of bare blocks for which a test had to be implemented. We decided to employ full scan design for the individual blocks. As for any firm or hard cores the scan chain configuration was taken as fixed later on. After scan chain insertion ATPG was performed for each individual block. On-chip memories were assigned a zero solid /one solid sequence of patterns to test them. The results are shown in Table I. Column 1 shows the names of the blocks of LEON as shown in Figure 5. Columns 2 and 3 show the number of primary inputs and outputs of each block. Column 4 shows the size of the combinational part of each unit in combinational area units (CAU) reported by the logic synthesis tool and Column 5 the number of flip-flops of each unit. The number of test patterns generated for each block is shown in Column 6.

Block Name	**PI**	**PO**	CAU	**FF count**	**Test** patterns
Integer Unit	139	273	2196	592	134
Data Cache	221	215	824	167	74
Inst. Cache	192	171	361	53	57
Mem. Ctrl.	145	155	578	216	268
Timers	32	27	516	127	51
UART2	20	18	228	94	36
UART1	20	18	228	94	34
I/O Port	63	80	220	104	31
IRQ Ctrl.	56	34	236	64	68
APB Bridge	301	377	299	13	29
AHB Master	145	191	181	9	15
Acache	157	161	77	13	16
Reg. File[1]	58	64	–	–	512
Cache Mem.[2]	140	114	–	–	1024

TABLE I
ATPG RESULTS FOR THE SET OF CORES.

C. Test Wrappers

Test wrappers isolate each block from its system environment during test. A test wrapper was inserted manually at RT level by modifying the VHDL source code. The selected architecture was based on the IEEE P1500 wrapper model [16, 17]. For each core this model was only partially implemented as not all the functionality provided by the P1500 was necessary for concurrent test purposes. That means that each core wrapper only supports two of the six modes proposed by the standard IEEE P1500:

- *Normal operation mode,* in which the wrapper acts as a transparent interface between the core logic and the system chip.
- *Internal test mode,* either parallel or serial internal test, depending upon core necessities. In this mode the necessary core isolation is provided as well as the internal core test through the input and output TAMs, implementing the necessary parallel to serial or serial to parallel test data

conversion to cope with the relationship between core terminals and chip pins.

In the test of the Leon SoC platform we focused on the test of the blocks itself, since they usually dominate the test application time. No external test modes were implemented for interconnection test.

After wrapping the cores (including on-chip memories), the required TAM width and required test application time is known. The data are shown in Table II. Column 1 shows the name of the block, Column 2 the required TAM width, which is the total number of scan chains and wrapper chains of the block. In Column 3 the number c of clock cycles necessary to run the test is given. This number is the sum of the number of scan cycles and capture cycles

$$c = ((N + 1) * SCAN_{cycles}) + N,$$

where N is the number of patterns that the test set for a given core contains and $SCAN_{cycles}$ is the maximum number of necessary clock cycles to scan in or scan out a test pattern of the wrapped core.

Block name	TAM Width	Test cycles
Integer Unit	20	6344
Data Cache	5	6374
Ins. Cache	1	14267
Mem. Ctrl.	8	14794
Timers	5	1715
UART2	3	1775
UART1	3	1679
I/O Port	4	1695
IRQ Ctrl.	2	4484
APB Master	1	2879
AHB Master	1	3215
Acache	1	2974
Reg. file	12	4102
Cache mem.	14	13323

TABLE II
EXPERIMENTAL RESULTS FOR THE WRAPPED CORES

D. Concurrent Test

Concurrent test requires, that the blocks are accessible from the chip pins by implementing a test access mechanism (TAM). A test controller was implemented, which scheduled the test of the blocks.

For each block of LEON we configured a separate port in the ATE. That implies that each core may be tested with its own pattern file and with its own frequency (which was not required for the LEON platform). 40 pins were available to test the SoC. Half of these were used for scan in, the other half were used for scan out.

1) TAM Design: The TAM transports test patterns on-chip. When selecting the TAM architecture for a core-based design the system integrator can tradeoff the TAM bandwidth (transport capacity) against area costs of the TAM. In our case study we maximized TAM bandwidth (i.e. minimized test time application) by matching the 40 externally accessible pins of the chip. This results in twenty pins available to apply test data through the input TAM and twenty pins to transport captured responses.

In a multiplexing TAM architecture every core gets assigned all available TAM bandwidth, i. e. all cores are connected to all scan inputs and all scan outputs of the IC [20]. Only one core may be tested at a time. In the distribution TAM architecture each core gets assigned its private TAM part and all cores may be tested concurrently [20]. Figure 6 sketches the selected TAM architecture, which is a mixture between a multiplexing and a distribution architecture.

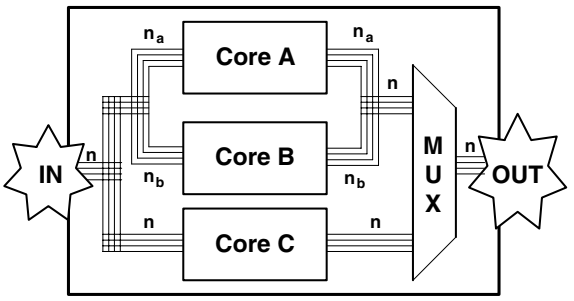

Fig. 6. Multiplexed Distribution TAM Architecture [20]

The pure distribution architecture maximizes concurrency, since the entire set of cores is tested concurrently. This architecture was not possible due to the constraint of total available bandwidth. The implemented concurrent test contained some sequential stages containing one or more cores (depending on bandwidth allocation necessities for the individual cores) in the same stage.

2) Test Scheduling: The goal of test scheduling is to maximize throughput by concurrently addressing as many cores as possible at every point in time without exceeding the constraint of global available bandwidth, i. e. twenty pins for test input stimuli and twenty pins for test responses capture. Test scheduling was performed manually in the case study. A *test controller* with 8 states was designed and implemented on-chip to perform the desired core test schedule. The test controller will set up the wrapper control inputs of each core properly thus establishing which core(s) is(are) under test at every moment during test execution. The selected test schedule for the set of cores is shown in Figure 7. The x axis shows the total TAM bandwidth usage, where 100% are 20 signals. The y axis shows the test application time in clock cycles. One to five cores are tested

[1] The register file was a 4Kbyte embedded RAM built from specific library cells for which no area information was provided by the synthesis tool.

[2] The same applies for the 6Kbyte embedded cache memory.

concurrently and throughput and test controller complexity are optimized. Introducing more controller states allows to remove the "idle time" between cache memory and memory controller.

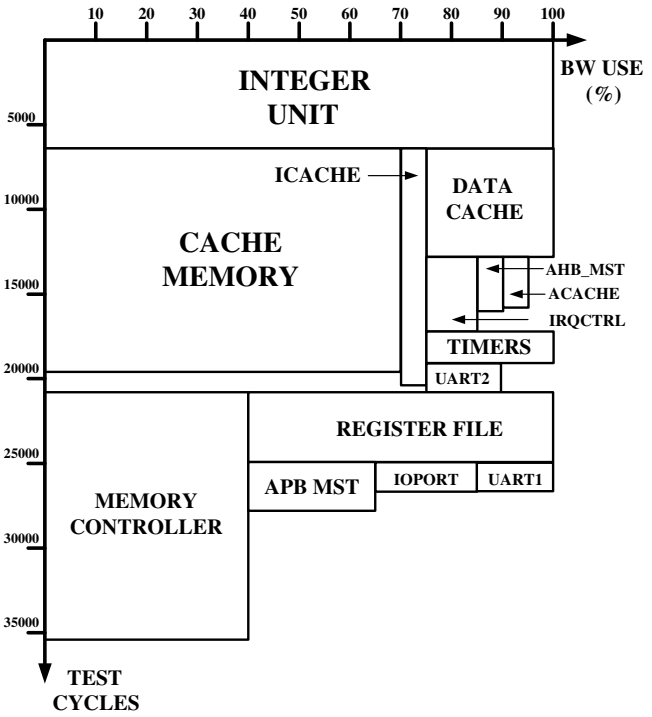

Fig. 7. Final test schedule

E. Testing Silicon on the Agilent 93000

The design was mapped on an FPGA and, after some manual manipulations to adapt the prototyping board to the tester loadboard, the global SoC test was run on the Agilent 93000 tester. This SoC tester was chosen for the case study, because it's Test-Processor-per-Pin architecture provides all capabilities to meet the tester requirements for concurrent testing as outlined in section II-A. The tester's programming environment for concurrent testing supports to freely assign every tester pin to an arbitrary number of independent ports, which is a prerequisite for efficiently testing core-based designs [12]. For each block of the platform, a tester port was allocated and the parameters such as the test frequency and the pattern file were customized on a block basis. Experimental results about testing times will be presented in Section V.

IV. Test Flow

LEON does not come with any DfT or test patterns. In order to perform a concurrent test for each block, scan chains were inserted and configured as shown in the previous section. For each block test patterns were generated. Then wrappers and the TAM were inserted at RT level.

A. General Flow

The general design flow for the implementation of the concurrent test approach on the Leon SoC is shown in Figure 8.

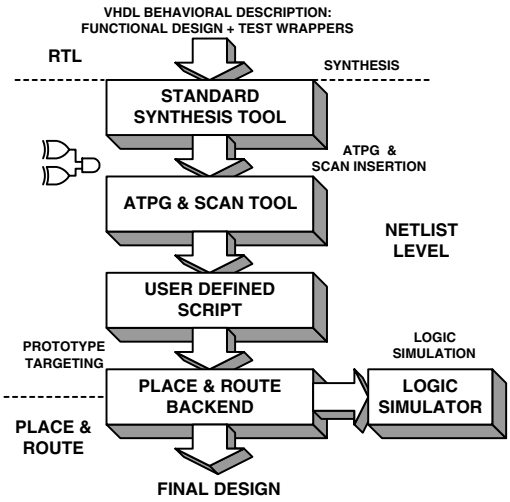

Fig. 8. Initial Design Flow.

In the first step all the necessary test wrappers, the TAM architecture and the test controller adding the necessary pins to the top entity file have to be codified in VHDL at register transfer level. Logic synthesis should produce the first system netlist. In the netlist of each block scan chains were inserted, and the necessary test patterns were generated using a commercial test insertion tool.

Finally, the steps corresponding to the physical domain such as place and route were performed with physical backend for the FPGA flow so that at the end we had the final design ready to be downloaded in the evaluation board. The wrapper's functionality as well as the correct operation of the test controller have been validated by a gate level simulator. Once the whole design was successfully simulated and no more modifications were needed the last step was to place the device on the real ATE and perform its concurrent test following the designed schedule.

B. Manual Interaction

There were caveats that induced to manually interact with the design flow. These problems can be mainly categorized as follows:

- **Unused bus signals:** the AMBA bus has a standardized set of signals in order to allow to plug predesigned and prevalidated cores in the LEON SoC platform. Even if the core does not use all bus signals, they are included in the core interface definition and are removed during logic synthesis. If these signals are wrapped:
  - Additional wrapper cells are needed

– Logic and interconnects cannot be removed during logic optimization, because it would change the behavior of the test wrapper when a pattern is shifted through the wrapper cells.

This may increase the area cost for concurrent test significantly and decrease yield due to unused but tested logic on the chip. In the wrapper design we kept track of the unconnected inputs and avoided with this manual procedure unnecessary wrapper cells. This task could be avoided if these cells would be tagged and a logic optimization tool removing these cells and the unused logic as it is done without a test wrapper.

- **Abstraction level related problems:** the designer decided to instantiate the wrappers at RTL because usually this implies less impact on the design performance than if the instantiation was done at netlist level. But the wrapper relies on a "DfT-ready" core and this happens to be true later on the design flow. For that reason some of the core to wrapper connections (e.g. scan in/out core terminals with wrapper TAM terminals) had to be implemented at netlist level. A script was developed which was inserting the connections. Alternatively, the complete wrapper could have been inserted at netlist level.

V. Results and Experiences

The main impacts of concurrent test on the design and test of the Leon SoC platform have been gate count costs, test application time reduction, and ease of core test integration.

A. Gate Count Costs

The area penalty introduced by the concurrent test approach is due to the following elements:

- The costs in terms of additional gates introduced by the test controller can be neglected in the total gate count. In our case, we implemented it as a finite state machine with eight different states, thus resulting in three additional flip-flops and some combinational gates. In a real SoC this is more than acceptable, as we are talking about designs containing several millions of transistors in the same silicon die.
- The costs caused by the wrappers was very high. It approximately tripled the total design size. Table III[3] shows that this increment in the design size was mainly due to instantiations of the wrappers for blocks, in which the wrapper logic was several times larger than the block logic. Column CAU shows the size of the blocks without wrappers in area units (AU) reported by the logic synthesis tool and Column WCAU the size of the blocks with wrappers. Column IF shows the incremenation factor. The costs may be reduced by clustering several blocks in a single wrapper. These costs are not specific for concurrent testing, but it is present in any core based SoC which isolates the cores from the system during test [16, 17, 21, 22].

[3]Statistics were not provided by the synthesis tool for the embedded memory cores.

Block name	CAU	WCAU	IF
Integer Unit	2929	4764	1.62
Data Cache	998	2946	2.95
Ins. Cache	424	2060	4.85
Mem. Ctrl.	1127	2329	2.06
Timers	686	933	1.36
UART2	351	504	1.43
UART1	351	502	1.43
I/O Port	325	903	2.77
IRQ Ctrl.	301	665	2.21
APB Master	294	3287	11.18
AHB Master	112	1558	13.91
Acache	97	1376	14.18
Total	7995	21827	2.73

TABLE III
Wrapper Ared Costs

- The number of interconnections reported by the physical design backend for the FPGA increased for the modified LEON design by a factor of 2.7. This is roughly the same number as the gate count increase for the test wrappers. Thus the additional wiring for the multiplexed-distribution TAM scheme may be neglected.

B. Test Application Time Reduction

Since the test assumption was that all blocks need a different test port, with a traditional tester only a multiplexing TAM would have been possible and all cores would be tested serially. This would result in a test application time of 79620 clock cycles, which is the sum of the cycles in the last column in Table II. On an SoC tester and prepared for concurrent testing Leon required a test application time for its blocks of 35,486 clock cycles with the described design modifications. Thus, we have reduced the test time by 44134 clock cycles, which is a reduction of 55,43 % compared to the serial test application time.

C. Core Test Integration

The integration of the core test into the system test was straight forward. It was sufficient to load the pattern file for each core onto the tester, no software conversion was necessary to merge the core tests into one big test pattern file. Only a couple of little scripts were developed to adapt the ASCII output produced by the ATPG tool to the ASCII input format required by the ATE software.

D. Tool Requirements

Implementing concurrent testing for the LEON SoC platform would be significantly easier if commercial EDA tools would fulfill the following requirements:

- Automatic selection of cores which are clustered into one block which is wrapped in order to avoid large area costs for the insertion of test wrappers
- Automatic insertion of a standardized wrapper at RT or netlist level. Unused signals removal and potential wrapper modifications should be done by the logic synthesis or a netlist postprocessing tool
- Selection of the TAM architecture and insertion of the TAM should be automated
- Tools for test scheduling, which were not available at the time the case study had been conducted
- Assigning ports in a way that they may share resources like clock signals could further improve the results

The past three items are not independent. So the selection of a TAM architecture in which two cores share resources may prohibit that they are tested concurrently. Furthermore, the integration of the scan chain design of soft cores into the optimization process for TAM, scheduling, and port assignment would be desirable.

VI. Conclusions

A concurrent test approach was implemented for an existing SoC platform. The implementation requirements for concurrent test are a subset of any core based test approach, like the IEEE P1500 standard for embedded core test [16, 17, 21, 22] or proprietary methods like PhilipsTestRail / TestShell / TestControlMechanism [15]. The area costs for implementing a concurrent test strategy for systems which have a core based test strategy is very low. Using the concurrent testing functionality of an SoC tester reduced test application time significantly. Test generation for concurrent test is currently a time consuming task due to a lack of tool support. With the efforts to define a standardized test interface between core and system, appropriate EDA tools may be implemented to automate the concurrent test methodology, fulfilling the short time to market requirements for today's SoCs.

References

[1] M. Birnbaum and H. Sachs, "How VSIA Answers the SOC Dilema," *IEEE Computer*, vol. 32, pp. 42–50, June 1999.

[2] D. Flynn, "AMBA: Enabling Reusable On-Chip Design," *IEEE Micro*, pp. 20–27, July 1997.

[3] W. Peterson, "Design philosophy of the wishbone soc architecture," 1999.

[4] D. Bhattacharya, "Hierarchical Test Access Architecture for Embedded Cores in an Integrated Circuit," in *Proceedings of the VLSI Test Symposium (VTS)*, pp. 8–14, IEEE, 1998.

[5] M. Nourani and C. Papachristou, "Parallelism in Structural Fault Testing of Embedded Cores," in *Proceedings of the VLSI Test Symposium (VTS)*, pp. 15–20, IEEE, 1998.

[6] I. Ghosh, N. K. Jha, and S. Dey, "A Low Overhead Design for Testability and Test Generation Technique for Core-Based Systems," in *Proceedings of the IEEE International Test Conference (ITC)*, pp. 50–59, October 1997.

[7] I. Ghosh, S. Dey, and N. K. Jha, "A Fast and Low Cost Testing Technique for Core-based System-on-Chip," in *Design Automation Conference*, pp. 542–547, June 1998.

[8] L. Whetsel, "Addressable Test Ports: An Approach to Testing Embedded Cores," in *Proceedings of the IEEE International Test Conference (ITC)*, pp. 1055–1064, IEEE Computer Society Press, September 1999.

[9] M. Nourani and C. Papachristou, "Structural Fault Testing of Embedded Cores Using Pipelining," *Journal of Electronic Testing Theory and Applications (JETTA)*, vol. 15, pp. 129–144, August/October 1999.

[10] E. Larsson and Z. Peng, "An Integrated System-On-Chip Test Framework," in *Proceedings of the Design Automation and Test in Europe (DATE)*, pp. 138–144, 2001.

[11] Y. Zorian, E. Marinissen, and S. Dey, "Testing Embedded-Core-Based System Chips," in *Proceedings of the IEEE International Test Conference (ITC)*, pp. 130–143, IEEE, 1998.

[12] M. Goto and K.-D. Hilliges, "The DFT-Age ATE Architecture - The Multi-Port ATE," in *SEMICON – SEMI Technology Symposium (STS)*, (Chiba, Japan), pp. 5–82 – 5–91, 2000.

[13] L.-T. Wang and M. Fischer, "Concurrent testing races to catch up with SoCs," *Integrated Communications Design (ICD)*, March 2001.

[14] M. Fischer, "Concurrent Test - A Breakthrough Approach for Test Cost Reduction," in *European Manufacturing Test Conference (EMTC)*, 2001.

[15] E. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousberg, and C. Wouters, "A Stuctured and Scalable Mechanism for Test Access to Embedded Reusable Cores," in *Proceedings of the IEEE International Test Conference (ITC)*, (Washington, DC), pp. 284–293, IEEE Computer Society Press, October 1998.

[16] E. J. Marinissen, Y. Zorian, R. Kapur, T. Taylor, and L. Whetsel, "Towards a Standard for Embedded Core Test: An Example," in *Proceedings of the IEEE International Test Conference (ITC)*, pp. 616–627, IEEE, 1999.

[17] E. J. Marinissen, "On Using IEEE P1500 SECT for Test Plug-n-Play," in *Proceedings of the IEEE International Test Conference (ITC)*, (Atlantic City, NJ), pp. 770–777, IEEE, IEEE, 2000.

[18] J. Gaisler, "LEON Web Site." http://www.gaisler.com/, 2002.

[19] SPARC International, Inc, http://www.spart.org/standards/, *The SPARC Architecture Manual Version 8*, 1992. Revision SAV080SI9308.

[20] J. Aerts and E. J. Marinissen, "Scan Chain Design for Test Time Reduction in Core-Based ICs," in *Proceedings of the IEEE International Test Conference (ITC)*, pp. 448–457, IEEE, 1998.

[21] Y. Zorian, "Test Requirements for Embedded Core-based Systems and IEEE P1500," in *Proceedings of the IEEE International Test Conference (ITC)*, pp. 191–199, 1997.

[22] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing Embedded-Core-Based System Chips," *IEEE Computer*, vol. 32, pp. 52–60, June 1999.