Testing of SoCs with Hierarchical Cores: Common Fallacies, Test Access Optimization, and Test Scheduling

Sandeep Kumar Goel, *Senior Member, IEEE*, Erik Jan Marinissen, *Senior Member, IEEE*, Anuja Sehgal, *Member, IEEE*, and Krishnendu Chakrabarty, *Fellow, IEEE*

Abstract—Many system-on-chip (SoC) integrated circuits today contain multiple hierarchy levels for both design and test. Hierarchy imposes constraints on the manner in which tests must be applied to “parent” cores and their “child” cores. However, most prior work on wrapper design, test access mechanism (TAM) optimization, and test scheduling is hierarchy oblivious, i.e., these techniques treat all cores in an SoC as if they are at the same level of hierarchy. We first show that the test architecture, consisting of wrappers and TAMs, and the corresponding test schedule designed for nonhierarchical SoCs are not valid for SoCs with hierarchical cores. Next, we present two approaches for efficient testing of SoCs with hierarchical cores. In the first approach, the problem is solved by extending a conventional wrapper design; this approach leaves full flexibility for TAM optimization and test scheduling. The second approach is based on a modified wrapper design for parent cores that operates in two disjoint modes for testing of parent and child cores. This approach has an impact on the test architecture and corresponding schedule. We show how an existing test architecture design algorithm can be adapted for use with both approaches. Experiments with the ITC’02 SoC Test Benchmarks show that the first approach offers lower test application times, while the second approach incurs less area costs.

Index Terms—SoC test, TAMs and wrappers, hierarchical SoCs, test scheduling, test architecture design.

1 INTRODUCTION

THE integration of a complete electronic system on a single chip is now commonplace. This achievement can be attributed to advances in semiconductor process technology and design methods, which have been fueled by the need for high performance, low power, and short time-to-market. A System-on-Chip (SoC) typically integrates a heterogeneous mix of digital logic, embedded memories, and analog blocks. Ever-increasing SoC complexity and diminishing product cycles are resulting in the widespread use of predesigned and preverified third-party cores such as CPUs, DSPs, media coprocessors, memories, and mixed-signal blocks.

Due to functional and performance requirements, modern SoC designs are not limited to only one level of design and test hierarchy (SoC and cores); instead, they contain multiple levels of hierarchy. For example, Dutta et al. [1] and Goel et al. [2] describe SoCs for digital video, for which the design is partitioned into design and test units called

chiplets, which in turn consist of multiple cores. Design units that contain other cores are referred to as *hierarchical cores*, while cores that do not contain other cores are referred to as *flat cores*. A hierarchical core is also called a *parent core*, while the cores that are at one level below and embedded in a parent core are referred to as *child cores*. In turn, a child core itself can be a parent core for the cores at deeper levels of hierarchy.

To simplify and speed up test generation and to enhance test reuse, modular testing of SoCs is strongly advocated [2], [3]. In modular testing, all embedded cores are tested independently from each other. This approach is mandatory for embedded nonlogic components such as memories and analog modules, as well as for black-box third-party cores [3]. Modular testing requires an on-chip test access infrastructure [4], which consists of test access mechanisms (TAMs) and test wrappers. TAMs [5], [6] transport test stimuli and responses between SoC pins and core terminals, and vice versa, while a test wrapper [5], [6] is a thin shell around a core that forms the interface between the core and its SoC environment. The wrapper connects the core terminals to the rest of the SoC and the TAM. It provides switching between various modes of operation, such as normal (functional) mode, core-internal (inward-facing) test mode, and core-external (outward-facing) test mode. Wrappers are designed such that the core I/O width is adapted to the available TAM width, e.g., by means of serial-to-parallel or parallel-to-serial conversion [7]. IEEE Std. 1500 [8] defines a standardized but scalable wrapper architecture.

Modular testing is generally known to reduce the test length, i.e., the SoC test application time and the vector memory required on the tester. Various wrapper design [7],

- S.K. Goel is with LSI Corporation, 1621 Barber Lane, M/S AD222, Milpitas, CA 95035. E-mail: sandeep.goel@lsi.com.
- E.J. Marinissen is with IMEC vzw, Kapeldreef 75, B-3001 Leuven, Belgium. E-mail: erik.jan.marinissen@imec.be.
- A. Sehgal is with NVIDIA, 2701 San Tomas Expressway, Santa Clara, CA 95050. E-mail: anuja.banerjee@nvidia.com.
- K. Chakrabarty is with the Department of Electrical and Computer Engineering, Duke University, Box 90291, 130 Hudson Hall, Durham, NC 27708. E-mail: krish@ee.duke.edu.

Manuscript received 26 Sept. 2007; revised 17 June 2008; accepted 7 July 2008; published online 8 Sept. 2008.

Recommended for acceptance by P. Girard.

For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2007-09-0484. Digital Object Identifier no. 10.1109/TC.2008.169.

[9], [10] and test access infrastructure optimization algorithms [11], [12], [13], [14], [15], [16], [17], [18], [19] have been described in the literature. Unfortunately, all these methods unrealistically assume that there is no hierarchy inside the embedded cores. Even if the benchmark SoCs used contain hierarchical cores, these optimization techniques treat all cores in the SoC as if at the same level of hierarchy. Hierarchy imposes constraints on the manner in which tests must be applied to parent cores and their child cores. Wrappers, TAMs, and test schedules created as if the SoC was nonhierarchical are typically not valid for SoCs with hierarchical cores. Therefore, test solutions proposed by the methods in [7], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], and [19] are not directly applicable to real-life SoCs. There is a need for wrapper design and TAM optimization techniques that can handle hierarchical SoCs.

In this paper, we describe two approaches for efficient testing of SoCs with a mix of hierarchical and nonhierarchical cores. In the first approach, we extend the existing wrapper architecture in such a way that all constraints imposed by the hierarchy are satisfied and full flexibility is provided to a test architecture design algorithm for an SoC with hierarchical cores. In this way, an efficient test schedule with (near-) optimal test length is obtained for such an SoC. In the second approach, we propose a new hierarchy-aware wrapper architecture for parent cores that has two disjoint test modes for testing of parent and child cores. We show how an existing test architecture design algorithm can be adapted to utilize the proposed approaches. The first approach has the advantage of lower test application time, while the area cost for the second approach is lower compared to the first method.

The remainder of this paper is organized as follows: Section 2 provides an overview of related prior work. In Section 3, we discuss the testing of SoCs with hierarchical cores, and why test architectures and test schedules made for flat cores are invalid for hierarchical cores. One of the root causes lies in the wrapper cell implementation, which is described extensively. In Section 4, we define a classification of problem definitions for hierarchical SoCs and provide an overview of our two solution approaches. Sections 5 and 6 present details of both approaches. Experimental results for the ITC'02 SoC Test Benchmarks are presented in Section 7. Finally, Section 8 concludes this paper.

2 RELATED PRIOR WORK

Hierarchical cores can have multiple levels of hierarchy. They contain embedded cores, which in turn can contain other embedded cores at deeper levels of hierarchy. Therefore, it is quite natural to use a recursive model to describe hierarchical cores. A generic recursive model was presented in [20]. According to this model, the hierarchy present in the test view of an SoC can be easily represented by its test hierarchy tree, where nodes represent cores and an edge between two nodes represents the hierarchical relation between the corresponding cores. All leaf nodes in a test hierarchy tree represent nonhierarchical (i.e., flat) cores, while the root node represents the top-level test entity, either core or SoC. The depth of a node represents the level of the corresponding core in the test hierarchy. A node (core) at

Fig. 1. Example of (a) a hierarchical core and (b) its test hierarchy tree.

depth n in the tree is called a *parent* node (core) with respect to the nodes (cores) that are connected to it and are at depth $n + 1$. Conversely, nodes at depth $n + 1$ are called *child* nodes with respect to the node that is at depth n and connected to these nodes. Parent nodes may have multiple child nodes, which in turn can be parent nodes for other nodes.

Fig. 1a shows an example of a hierarchical core *A*. Core *A* contains three child cores *B*, *C*, and *D*, of which *B* and *D* are also hierarchical cores. Core *B* contains only one child core *E*, while core *D* contains two child cores *F* and *G*. Core *G* itself contains a child core *H*. Therefore, core *G* is a child core of core *D* but the parent core for core *H*. Fig. 1b shows the test hierarchy tree for core *A*. Core *A* is at the top level (depth 0), while core *H* is at the lowest level (depth 3) in the test hierarchy tree.

Most prior work on wrapper/TAM optimization for SoCs has assumed a nonhierarchical test infrastructure [11], [12], [13], [14], [15], [16], [17], [18], [19], [21], [22], [23], [24], [25], [26], [27]. In comparison, only a limited amount of work has been done on wrapper design and TAM optimization for SoCs with hierarchical cores. Recently in [28], [29], and [30], techniques for wrapper/TAM optimization for hierarchical SoCs have been explored. In [28], an existing hierarchy-oblivious TAM optimization approach is used to iteratively solve the problem of TAM optimization for hierarchical SoCs. However, in this approach, the constraints related to simultaneous testing of parent and child cores are ignored. In [29], a TAM design technique for hierarchical SoCs is presented in which the hierarchical cores are assumed to be hard wrapped cores. This approach requires large area cost due to the added registers for bandwidth matching, and it also requires synchronization of the clock signals. Recently, in [30], a test scheduling technique was presented with the objective to minimize the test application time while considering multiple constraints due to cross-core testing (testing of interconnections between cores), multiple test sets, hierarchical conflicts in SoCs, the sharing of the TAM, test power limitations, and precedence conflicts. Although hierarchical conflicts such as parallel testing of parent and child cores are considered, the requirement of access to child core wrapper while testing the parent core is not considered.

3 TESTING OF SoCs WITH HIERARCHICAL CORES

In this section, we describe wrapper cell designs and highlight constraints that arise for hierarchical cores.

Fig. 2. IEEE Std. 1500 basic wrapper cell WC_SD1_CII: (a) behavioral specification and (b) a sample gate-level implementation.

3.1 Wrapper Cell Implementation

Wrapper cell implementations typically put constraints on the scheduling of tests for hierarchical cores. This phenomenon is explained in this section, especially for commonly used IEEE Std. 1500 wrapper cells. IEEE Std. 1500 [8] is a global, industry-wide standard for core test wrappers. The standard only specifies wrapper behavior and does not prescribe a particular implementation. However, for individual wrapper cells, behavior and implementation have almost a one-to-one relation. IEEE Std. 1500 describes a basic wrapper cell but also allows for variants and extensions of both basic cell and variants.

Fig. 2a depicts the behavioral description of IEEE Std. 1500 basic wrapper cell WC_SD1_CII, while Fig. 2b shows a natural corresponding sample gate-level implementation. In the figure, CFI and CFO represent the functional input and output terminals, while CTI and CTO are the test input and output terminals.

A wrapper cell has three main modes: 1) transparent, 2) drive, and 3) capture. The *transparent* mode is the regular functional mode, in which data passes unhindered from functional input CFI to functional output CFO. The corresponding control settings of the two multiplexers in the basic IEEE Std. 1500 wrapper cell are $m0 = X$ and $m1 = 0$. The other two modes are test modes. In the *drive* mode, the wrapper cell shifts test stimuli in from test input CTI and delivers them at functional output CFO. The corresponding multiplexer control settings are $m0 = 1$ and $m1 = 1$. In the *capture* mode, the wrapper cell captures test responses from functional input CFI and shifts them out through test output CTO. The corresponding multiplexer control settings are $m0 = 0$ and $m1 = X$. Fig. 3 depicts these three modes; by means of thick lines, the figure highlights which nets are active during a particular mode.

In the INTEST mode of a wrapper, the wrapper cells at core inputs are in drive mode, while the wrapper cells at core outputs are in capture mode. In the EXTEST mode, these

Fig. 4. (a) Testability problem with the basic IEEE Std. 1500 wrapper cell. (b) A fully testable alternative wrapper cell.

roles are reversed, i.e., the input wrapper cells are in capture mode and the output wrapper cells are in drive mode. This implies that the basic IEEE Std. 1500 wrapper cell has a testability problem. The combination of INTEST and EXTEST does *not* cover the test of the CFI to CFO connection and the upper leg of the multiplexer in that path ($m1 = 0$), indicated by the dotted line in Fig. 4a.

IEEE Std. 1500 also allows for a variant of the basic wrapper cell, which does not suffer from the testability problem described above. This alternative wrapper cell is named WC_SD1_COI and is shown in Fig. 4b. In this wrapper cell, the captured data are not tapped off from CFI but from CFO (after multiplexer m1) instead. In this way, the functional path from CFI to CFO is exercised in the capture mode, and hence, the wrapper cell is fully tested in the combination of INTEST and EXTEST modes. Due to its good testability features, this cell is widely used in the industry [5], [7]. Note that for this popular wrapper cell, the drive and compare modes are mutually exclusive, due to the fact that the multiplexer m1 can only be in one position at a time. Consequently, a wrapper that contains this wrapper cell cannot execute its INTEST and EXTEST modes simultaneously.

3.2 Why Flat Schedules Are Invalid for Hierarchical Cores

The introduction of hierarchy in testing has repercussions on the design of the test access architecture and test scheduling. Testing of a core requires the core's wrapper to be in its inward-facing INTEST mode. Test stimuli are

Fig. 3. IEEE Std. 1500 basic wrapper cell configurations in different modes. (a) Transparent mode. (b) Drive mode. (c) Capture mode.

Fig. 5. Wrapper-based stimuli application and response capturing during the test of core *A*. (a) Flat core. (b) Hierarchical core.

applied at the core's input terminals via the wrapper cells on those inputs, while test responses are captured at the core's output terminals via the wrapper cells on those outputs. This is schematically depicted for a sample core *A* in Fig. 5a. In the case of testing a hierarchical core, in addition it is required that the wrappers of the child cores are in their outward-facing EXTEST mode. The output terminals of these child cores serve as inputs to the parent core, and hence, the wrapper cells on the outputs of the child cores are used to drive test stimuli into the parent core. Similarly, the input terminals of the child cores serve as outputs to the parent core, and hence, the wrapper cells on the inputs of the child cores are used to capture test responses from the parent core. This is schematically depicted in Fig. 5b, for example, parent core *A* and its child core *B*.

During test scheduling for hierarchical cores, the following constraints should be taken into account:

- Wrappers that are based on wrapper cell WC_SD1_COI cannot simultaneously be in INTEST and EXTEST modes. Executing a parent core test requires its child core wrappers to be in EXTEST mode, while executing the child core tests requires these child core wrappers to be in INTEST mode. Hence, parent INTEST and child INTEST cannot run simultaneously.
- Testing a parent core requires the utilization of not only the TAM that connects to the parent core's wrapper but also utilization of the TAM(s) that connect to the wrappers of its child cores. Many TAMs can only serve one core at a time.

Test schedules that do not explicitly take these constraints into account are likely to be invalid. Consider the test architecture of a sample hierarchical SoC shown in Fig. 6a. The SoC contains four cores, of which only core *A* is a hierarchical core and contains core *B*. The test architecture shown here comprises of two TAMs of widths $w1$ and $w2$, respectively. The TAM of width $w1$ connects to cores *C* and *B*, while the other TAM connects to cores *A* and *D*.

Figs. 6b and 6c show two schedules for the sample hierarchical SoC. The horizontal axis in these schedules represents test length (in clock cycles), while the vertical axis represents TAM width (in wires). The rectangles in the schedule denote the INTESTS for the various cores in the SoC. The schedule in Fig. 6b was put together as if all cores in the SoC are the same hierarchical level, as would be done in [11], [12], [13], [14], [15], [16], [17], [18], [19], [24], [25], [26],

Fig. 6. (a) Test architecture for a sample hierarchical SoC. (b) Hierarchy-unaware invalid test schedule. (c) Hierarchy-aware test schedule.

[27], and [28]. This schedule ignores the constraints for hierarchical cores and is invalid because it executes the tests for cores *A* and *B* in parallel.

Fig. 6c shows an example of a modified test schedule that respects the hierarchy present in the SoC. In this schedule, when core *A* is tested, its child core is put into EXTEST mode. It can be seen from Fig. 6c that the modified test schedule results in a large penalty in test length (+65 percent).

4 PROBLEM CLASSIFICATION AND OVERVIEW OF SOLUTION APPROACHES

The conventional problem in modular, core-based testing is to determine a TAM architecture and wrapper design for each core, such that the overall SoC-level test length is minimized. The test details of all cores and a maximum SoC-level number of TAM wires are provided; in [16], formal definitions of this problem are given for both cores with fixed scan chains (TADHM) and flexible scan chains (TADSM). It is the objective of this section to extend those problem definitions to SoCs with hierarchical cores and provide an overview of the two solution approaches proposed in this paper.

The test access infrastructure for an embedded core consists of three components: 1) the core-internal scan chains, 2) the core-external TAM, and 3) the test wrapper around the core, which connects the scan chains and TAM to each other. The parameters of these components (e.g., the number and length of the scan chains, and the width and architecture of the TAM) determine the test access architecture, the corresponding test schedule, and hence the required test length. The parameters of the test access components can either be given and fixed or flexible and to-be-fixed; we refer to that as respectively *hard* and *soft*. In a hierarchical setting, we have these three-tuples of test access components, both for the parent core (denoted as $P(T, W, S)$) and the child core ($C(T, W, S)$). With a three-tuple (T, W, S) , we denote the

Fig. 7. Sample implementations for the proposed wrapper cell. (a) Wrapper input cell. (b) Wrapper output cell.

three components TAM, wrapper, and scan chains, which can either be hard (h), soft (s), or don't care (x) (i.e., $T, W, S \in \{h, s, x\}$).

An example of a realistic design scenario is the one in which a parent core and everything in it (child scan chains, child wrapper, child TAM, and parent scan chains) are hard, while parent wrapper and parent TAM are still to be fixed and hence soft; the parent core could well be a microprocessor with embedded memories, for which the test integration tasks are to wrap it and connect it into an SoC-level test architecture. This setting is denoted as $P(s, s, h); C(h, h, h)$. Another realistic design scenario [31] is the one in which the top-level TAM width and architecture are determined first (hard), while all (parent and child) core-level scan chains and wrappers are still soft and to be adapted to it: $P(h, s, s); C(s, s, s)$.

The above classification yields $2^6 = 64$ possible problem settings, which are all slightly different. Fortunately, not all these problem settings are equally realistic. It is one of the functions of a core test wrapper to match the (possibly unequal) numbers of core-internal scan chains and TAM wires; hence, a design scenario in which the wrapper of a core is hard, while its scan chains and TAM are still soft ($P(s, h, s)$ or $C(s, h, s)$), is unlikely.

In this paper, we propose two alternative solutions that both result in valid test architectures and corresponding test schedules for SoCs with hierarchical cores:

1. **Modified Wrapper Cell (MWC) Design.** In this approach, we modify the wrapper cell design for all child cores, such that parallel testing of parent and child cores is possible. This approach is applicable only in a $C(x, s, x)$ problem setting, as the wrappers of child cores need to be modified. At the expense of extra silicon area required for the wrapper cells of the child cores, this approach allows full freedom to any test architecture design algorithm (including

hierarchical-oblivious algorithms) to obtain optimal test length.

2. **New Wrapper Architecture.** In this approach, we propose a new wrapper architecture for parent cores, with two disjoint test modes for the testing of parent and child cores. This approach is applicable only in a $P(x, s, x)$ problem setting, as the wrappers of parent cores need to be modified. At the expense of a longer test length, we avoid the silicon area cost of the first solution approach.

5 MODIFIED WRAPPER CELL

To allow parallel testing of both parent and child cores, we propose to modify the wrapper cells in the child core wrapper [32], [33]. Unlike the conventional wrapper cell, which is connected to only one TAM, the proposed wrapper cell is connected to the following two TAMs: 1) *child core TAM*, to serve the test data requirements for the child core and 2) *parent core TAM*, to serve the test data requirements for the parent core.

Fig. 7a shows a sample implementation for the proposed wrapper input cell. In this cell, there are two flip-flops; flip-flop FF 1 is again used to store test data for the child core test, while the newly added flip-flop FF 2 is used to store test data for the parent core test. In Fig. 7a, CFI and CFO represent the functional input and output terminals. CTI and CTO represent the test input and output signals corresponding to the child core TAM, while PTI and PTO represent the same for the parent core TAM. The terminal CFI is connected to the primary signal coming from the parent core. Similarly, terminal CFO is connected to the primary signal going to child core. Fig. 7b shows a sample implementation for the proposed wrapper output cells, respectively. Opposite to the wrapper input cell, in this cell, terminal CFI is connected to the primary signal coming from the child core and terminal CFO is connected to the primary signal going to the parent core.

Figs. 8 and 9 show the proposed wrapper input and output cells configured in various modes. The thick black line in the figure shows the active path in the corresponding mode. The INTEST mode is used to test the child core itself, while the EXTEST mode is used during the parent core test. Table 1 shows the setting of the multiplexers for the proposed wrapper cells in the various supported modes. Logic value "x" represents the *don't-care* term. From the table, one can see that the settings for the INTEST and EXTEST modes are compatible with each other. Hence, with this type of wrapper cell, a core can be configured in both

Fig. 8. Configuration of the proposed wrapper input cell in various supported modes. (a) INTEST shift. (b) INTEST normal. (c) EXTEST shift. (d) EXTEST normal.

Fig. 9. Configuration of the proposed wrapper output cell in various supported modes. (a) INTEST shift. (b) INTEST normal. (c) EXTEST shift. (d) EXTEST normal.

the INTEST and EXTEST modes at the same time. Therefore, the testing of parent and child cores can be done in parallel if they are connected to different TAMs.

5.1 Testability of the Proposed Wrapper Cells

The proposed wrapper input cell is fully testable as all nodes in the cell are fully controllable and observable. On the contrary, the wrapper output cell is not fully testable. It is due to the fact that the value on the output signal of multiplexer m5 cannot be observed during any test mode. In order to make the proposed wrapper output cell fully testable, an additional multiplexer can be added. Fig. 10 shows an example of the fully testable wrapper output cell.

To observe the value on the output signal of multiplexer m5, the newly added multiplexer mx should be set to logic value "0" in normal operation during the EXTEST mode. During this mode, the value on the output signal of the multiplexer m5 can be captured in the flip-flop FF 2 and then shifted out during the shift operation. For the rest of the modes, the setting of the multiplexer mx is not important and hence can be considered as *don't care* ("X").

5.2 Ordering of Wrapper Cells

In the proposed wrapper cells for the child core, the parent TAM also connects to the wrapper cells in the child core wrapper. Therefore, in a wrapper architecture that uses the proposed wrapper cells in the child core wrapper, the parent TAM is connected to the following elements:

- scan chains in the parent core,
- wrapper input cells connected to the parent core's functional input terminals,
- wrapper output cells connected to the parent core's functional output terminals,
- wrapper input cells connected to the child core's functional input terminals,

TABLE 1
Multiplexer Settings in the Proposed Wrapper Cells for Various Modes

Wrapper mode	Multiplexer settings					
	Wrapper input cell			Wrapper output cell		
	m1	m2	m3	m4	m5	m6
Transparent	X	1	X	X	1	X
InTest shift	1	X	X	1	X	X
InTest normal	0	0	X	0	X	X
ExTest shift	X	X	0	X	X	0
ExTest normal	X	X	1	X	0	X

- wrapper output cells connected to the child core's functional output terminals.

To minimize the test length for the parent core, we describe an optimal ordering for the above-mentioned elements in a TAM connected to the parent core. As described earlier, in order to test the parent core, we need to shift test stimuli into its scan chains, its wrapper input cells, and also to the wrapper output cells of its child core. Similarly, one needs to shift out test responses from its scan chains, its wrapper output cells, and also from the wrapper input cells of its child core. Fig. 11 shows the proposed optimal ordering of the various elements in a single TAM wire that is connected to the parent core.

In Fig. 11, boxes containing IDs Pa_i and Pz_j represent parent wrapper input and output cells, respectively. Similarly, boxes containing IDs Ca_i and Cz_j represent child wrapper input and output cells, respectively. As the scan chains take part both in applying and observing test data, they are in the middle of the wrapper cells. The wrapper input cells for the parent core together with the wrapper output cells for the child core are connected in front of the scan chains. Likewise, the wrapper input cells for the child core and the wrapper output cells for the parent core are connected after the scan chains.

Based on the above ordering, Fig. 12a shows the improved wrapper configured in the parent INTEST and child EXTEST modes. Fig. 12b shows the improved wrapper architecture configured in the child INTEST mode. In both figures, active connections are shown by thick black lines, while the inactive connections are shown by gray lines. As

Fig. 10. Sample implementation for the fully testable wrapper output cell.

Fig. 11. Ordering of various elements in a TAM connected to the parent core.

Fig. 12. Improved wrapper architecture configurations for parent and child core tests. (a) Parent core test. (b) Child core test.

far as the impact of new wrapper cell on the functional performance of the core is concerned, there is no difference as compared to the conventional wrapper cell. This is due to the fact that in the new wrapper cell, there is only one multiplexer in the functional path from CFI to CFO. The only drawback of the proposed wrapper cell is the area cost. Compared to the conventional wrapper cell, which only requires one flip-flop, the new wrapper cell requires two flip-flops and one/two additional multiplexers.

5.3 Test Architecture Design

A typical test architecture design algorithm generally contains two components: 1) a TAM partitioning and core assignment procedure and 2) a wrapper design routine. The TAM partitioning and core assignment procedure iteratively partitions the available TAM width over an optimal number of TAMs and assigns cores to these TAMs, such that the overall SoC test length is minimized. To calculate the test length for individual cores, it uses a wrapper design routine that designs the wrapper around a core for a given TAM width. In a typical wrapper design procedure [7], first scan chains are assigned to available TAM wires such that the maximum sum of scan lengths assigned to a TAM wire is minimized. Next, the input wrapper cells are distributed over TAM wires, such that the maximum scan-in time over all TAM wires is minimized. Finally, the output wrapper cells

are distributed over TAM wires, such that the maximum scan-out time over all TAM wires is minimized.

The integration of this solution into a typical test architecture design algorithm as described above requires a slight modification in the wrapper design procedure for parent cores only. There is no modification required in the TAM partitioning and core assignment procedures. Therefore, any test architecture design algorithm (originally developed for SoCs with flat cores) can be used for SoCs with hierarchical cores. The modification required in the wrapper design procedure is given as follows: While distributing the input wrapper cells in the parent core wrapper, the output wrapper cells in the child core wrapper should also be distributed over the TAM wires connected to the parent core. Similarly, while distributing the parent output wrapper cells, input wrapper cells in the child cores should also be distributed over the TAM wires connected to the parent core. Due to additional wrapper cells, the test length for a parent core can increase as compared to the case in which all cores are considered at the same hierarchical level. This can also result in a different overall test architecture.

6 NEW WRAPPER ARCHITECTURE

In this section, we present a new wrapper architecture for hierarchical cores and describe a procedure to design/optimize such a wrapper around a hierarchical core. The wrapper architecture presented here has two disjunct modes for testing of parent and child cores.

To design a wrapper around any core, whether hierarchical or nonhierarchical, we need to identify all types of terminals available at the core boundary. Based on the test access requirements, core terminals for nonhierarchical cores are classified as: functional-only, test data, and control terminals [7]. The functional-only terminals include functional input and outputs that require wrapper cells to apply test stimuli and observe test responses. The test data terminals are terminals directly connected to the scan chains of the core; hence, they do not require wrapper cells. Control terminals are used to apply control signals to the various components of the wrapper.

A parent core, in addition to the terminals mentioned above, has terminals that provide test access to its wrapped child cores. We refer to these terminals as CTAM as they also correspond to the terminals of the TAM connected to the child cores. Unlike the functional terminals at the parent core, CTAM terminals do not require wrapper cells at the parent core wrapper since the CTAM terminals connect only to the scan chains and/or the wrapper cells in the child core wrapper. These terminals also differ for the test data terminals at the parent core, as they operate in both INTTEST and EXTTEST modes. However, it should be noted that with the conventional wrapper cells used in our architecture, the INTTEST and EXTTEST modes for a core have to be time-multiplexed.

Fig. 13 illustrates an example of an unwrapped hierarchical core. The child cores in the sample hierarchical core are already wrapped and TAMED. In this example, the SI and SO terminals are the test data inputs and outputs, respectively, for the parent core scan chains, the FI and

Fig. 13. Example of a parent core with three wrapped child cores.

FO terminals are the functional inputs and outputs of the parent core, and similarly the $CTAM_I$ and $CTAM_O$ terminals are the CTAM inputs and outputs for the child cores. The parent core wrapper architecture shares several components in common with the IEEE Std. 1500 wrapper architecture described for flat cores in [34]. TAM ports (WPI and WSI), wrapper cells WBR, bypass register WBY, and wrapper control register WIR, as described in [34], have similar functionality in the parent core wrapper. However, additional components have been added to facilitate efficient test access to child cores embedded in the parent core. Also, the test modes of the parent core wrapper differ significantly from the test modes of the nonhierarchical core wrappers.

In the parent core, we have identified an EXTEST mode and two INTEST modes. The parent core EXTEST mode is similar to the EXTEST mode for the nonhierarchical cores. In this mode, the parent core terminals are used for core-external testing, i.e., to test the logic and circuitry outside the core itself. Only the wrapper cells in the parent wrapper are used in this mode and all scan chains in the parent core can be bypassed. In this mode, the child cores can be in INTEST or EXTEST mode, since they do not participate in the parent core wrapper functionality and are internal to the parent core.

The two INTEST modes that we have identified in a parent core are significantly different from the INTEST mode of a nonhierarchical SoC [35]:

1. **Parent INTEST mode (INTEST_P).** In this mode, parent core internal testing is done. Test data are scanned through the parent core's scan chains, the parent core's wrapper cells, and the child core's wrapper cells. In this mode, child cores need be in the EXTEST mode, as their wrapper output cells are required to apply test stimuli to the parent core, while their wrapper input cells are required to capture test responses from the parent core. As a result, test data have to be scanned through both the parent core and the child cores. Hence, the available

TAM wires have to be distributed between both the parent core scan chains and wrapper cells, as well as the child core TAM architecture. This was ignored in the prior work done on TAM optimization for hierarchical cores in [28]. For example, all the TAM wires were devoted to parent core testing during the INTEST mode of the hierarchical core.

2. **Child INTEST mode (INTEST_C).** In this mode, child core internal testing is done; all child cores are in INTEST mode. The parent core's wrapper elements can be in any mode of operation since the TAM inputs will be able to transport data to the child core's terminals regardless of the mode of operation of the parent core itself. Thus, in this mode, all the TAM wires can be utilized by the child cores for their INTEST testing.

It should be noted that the two INTEST modes described above have to be time-multiplexed, since both the modes require the TAM input terminals to be active. Here, we focus on designing a wrapper that is efficient, in terms of test length, for both INTEST modes. However, as the two modes are time-multiplexed, the top-level TAM wires available for the overall parent core testing at the parent core wrapper interface can be used in both modes. Multiplexers can be used to route the TAM wires to the child cores as well as the parent core scan chains and wrapper cells. These multiplexers can be controlled by control inputs that select the appropriate inputs depending on the mode of operation. We elaborate on this with the help of an example.

Fig. 14a shows an example of a parent core wrapper configuration when the core is in INTEST_P mode. In this wrapper, the available TAM wires are used to access the parent core scan chains, parent core wrapper cells, and the child core CTAM terminals. Fig. 14b shows the wrapper configuration for the same parent core in INTEST_C mode. In this configuration, the available TAM wires can be distributed between the CTAM inputs alone. Fig. 14c shows how the two wrapper configurations can be merged to form one wrapper, using multiplexers. The multiplexer select bits can be chosen appropriately to access the CTAM chains alone in the INTEST_C, or they can be chosen such that the TAM items participating in INTEST_P mode are selected.

An alternative approach to using the above described merging technique is to partition the available TAM width into two dedicated TAM partitions for the parent core and child core TAM architecture. In the INTEST_P mode, both the parent TAM partition and child TAM partition can be used, and in the INTEST_C mode, only the child TAM partition can be used. As a result, the TAM width available for child core testing in the INTEST_C mode is smaller than that available in our proposed approach.

In addition to the wrapper features described so far, optional bypasses can also be implemented that can be used in any of the three modes. In [36], two types of bypasses have been defined. The wrapper-wide bypass allows the bypass of an entire core. Single registers are used as bypasses to avoid delay effects, thus it takes one clock cycle to bypass the core; Fig. 13 shows core A equipped with a wrapper-wide bypass. A wrapper-wide bypass can exist outside or inside the wrapper. Also, scan chain bypasses can

Fig. 14. Parent core wrapper in (a) INTEST_P mode, (b) INTEST_C mode, and (c) Unified parent core wrapper.

be available that allow the scan chains of a core to be bypassed. Since the child cores are wrapped by the core provider, the use of bypasses in child cores would be left to the core provider's choice.

The problem of wrapper design for a hierarchical core can be considered as a twofold problem. Two wrappers can be designed independently for the two modes, i.e., INTEST_P and INTEST_C , and can be merged using multiplexers. These two problems together with their solutions are described below.

6.1 INTEST_P Mode Wrapper Optimization

The scan-in and scan-out times of a core should be minimized in order to optimize the overall test length of the core. For a hierarchical core, elements that participate in the scanning in and out of test patterns are

1. parent core wrapper input cells,
2. parent core scan chains,
3. parent core wrapper output cells, and
4. child core wrapper cells.

As the child cores in this case are already wrapped and TAMed, the child core wrapper cells and scan chains are connected to the child core TAM CTAM. Two or more CTAM wires (also referred as CTAM chains) can be daisy-chained at the parent level to minimize the test length for the parent core test.

The information about the number of scan chains and their lengths at the parent core level is provided by the core provider. The number of wrapper input and output cells is equal to the number of functional inputs and outputs, respectively. Also, the information about the scan lengths and scan-in and scan-out times of the child cores can be obtained from the core provider in the form of *test protocols*. Test protocols, provided for each child core by the core provider, carry information about the test stimuli and the scan times of the child core [37]. The sum of the number of wrapper cells and the length of the scan chains connected to a TAM chain is referred to as the *scan length* of that TAM chain. Thus, given the scan chain lengths, number of input and output wrapper cells at the parent core level, and the scan lengths of the child cores, we have all the information required to determine the maximum scan-in and scan-out times of the parent core in INTEST_P mode configuration. In the INTEST_P mode, we use the total scan lengths of the CTAM chains, instead of the scan-in and scan-out lengths, to calculate the overall test length of the core.

This assumption may increase the overall test length of the core negligibly; however, it reduces the complexity of the problem. Thus, with this information, we can proceed to define and solve the wrapper design problem for parent cores in the INTEST_P mode configuration.

In many practical cases, the number of wrapper cells and scan chains (referred to as TAM items) is much larger than the number of available external TAM wires. In such cases, the set of TAM items has to be partitioned into a number of subsets equal to the number of available TAM wires. The partitions should be made such that the maximum scan-in and scan-out times of the parent core are minimized. The wrapper design problem for the parent core in the INTEST_P mode can now be formalized as a partitioning problem as follows:

Problem 1 (wrapper design in INTEST_P mode). *Given a set $WI = \{WI_1, WI_2, \dots, WI_x\}$ of wrapper input cells, each wrapper input cell has a length $l(WI_i) = 1$. Given a set $S = S_1, S_2, \dots, S_y$ of parent core internal scan chains, scan chain S_i has a length $l(S_i)$. Given a set $WO = \{WO_1, WO_2, \dots, WO_z\}$ of wrapper output cells, each wrapper cell has a length $l(WO_i) = 1$. Given a set $S_c = \{S_{c,1}, S_{c,2}, \dots, S_{c,v}\}$ of CTAM scan chains, each scan chain has a length $l(S_{c,i})$. Furthermore, given a set of identical w TAM wires, we define for any $X \subset WI \cup S \cup WO \cup S_c$, $l(X) = \sum_{x \in X} l(x)$. A TAM partition is a partition $\mathcal{P} = \{P_1, P_2, \dots, P_w\}$ of $WI \cup S \cup WO \cup S_c$ into w disjoint sets, one for each TAM wire. We define input set $IN_i = P_i \setminus WO$. Likewise, we define output set $OUT_i = P_i \setminus WI$. The scan-in length for TAM partition $\mathcal{P}$ is defined by $si(\mathcal{P}) = \max_{1 \leq i \leq w} l(IN_i)$. The scan-out length for TAM partition $\mathcal{P}$ is defined by $so(\mathcal{P}) = \max_{1 \leq i \leq w} l(OUT_i)$.*

Objective. Find an optimal TAM partition $\mathcal{P}^*$ such that the overall test length of the core is minimized, i.e., $\mathcal{P}^*$ satisfies $\max(si(\mathcal{P}^*), so(\mathcal{P}^*)) \leq \max(si(\mathcal{P}), so(\mathcal{P}))$ for all partitions $\mathcal{P}$ of $WI \cup S \cup WO \cup S_c$.

The above problem is similar to the *partitioning of TAM chain items* (PTI) problem described in [7]. TAM chains described in [7] contain a subset of the set of parent TAM items, since they do not include child core scan chains. The PTI problem has been shown to be $\mathcal{NP}$ -hard in [7]. Thus, Problem 1, as described above, is also an $\mathcal{NP}$ -hard problem.

To solve Problem 1, we use a three-step approach similar to the one described in [7]:

1. Assign the parent core internal scan chains S and the child core scan chains S_c to w TAM chains, such that

the maximum sum of scan lengths assigned to a TAM chain is minimized. The resulting partition is named as $\mathcal{P}_S$.

2. Assign the wrapper input cells in $\mathcal{WI}$ to w TAM chains on top of $\mathcal{P}_S$, such that the maximum scan-in times of all w TAM chains is minimized.
3. Assign the wrapper output cells in $\mathcal{WO}$ to w TAM chains on top of $\mathcal{P}_S$, such that the maximum scan-out times of all w TAM chains is minimized.

Step 1 described above can be formalized as the *Partitioning of Scan Chains* (PSC) problem as described in [7] and can be solved using the Largest Processing Time (LPT) algorithm as described in [7]. As wrapper cells are of 1-bit size, their distribution over TAM chains (Steps 2 and 3) is trivial and it can be solved optimally.

6.2 INTEST_C Mode Wrapper Optimization

Next, we proceed to define the wrapper design problem for the hierarchical core in INTEST_C mode. In this mode of operation, the test stimuli have to be transported to the child cores only; hence, all the TAM wires available at the parent core wrapper interface can be utilized for child core testing.

Problem 2 (wrapper design in INTEST_C mode). *Given a set of CTAM chains M and child cores C . For each child core $c \in C$, the number of test patterns p_c , total scan length $sl_{c,k}$, scan-in time $si_{c,k}$, and scan-out time $so_{c,k}$ on k chain ($k \in M$) are given. Furthermore, we are given a number w that represents the maximum number of parent-core-level TAM wires available for testing.*

Objective. Determine a wrapper design for parent core such that the overall test length (in clock cycles) required to test all child cores is minimized and the number of TAM wires used for child core testing does not exceed w .

For example, in Fig. 14, $|M| = 6$. The total test length of testing all the child cores depends on the number of available TAM wires w and the child core TAM architecture. Let us consider two cases: 1) the number of available TAM wires is greater than or equal to the number of CTAM chains and 2) the number of available TAM wires is less than the number of CTAM chains.

Case 1: $w \geq |M|$.

If the number of available TAM wires w exceed the number of CTAM chains in the child core TAM architecture, then every CTAM chain can be connected to a separate TAM wire at the parent level. In this case, the total test length t_c for a core c can be written as

$$t_c = (1 + \max\{si_c, so_c\}) \cdot p_c + \min\{si_c, so_c\}, \quad (1)$$

where si_c and so_c are the maximum scan-in and scan-out times of core c and can be defined as follows: $si_c = \max_{k=1}^{|M|} \{si_{c,k}\}$, and $so_c = \max_{k=1}^{|M|} \{so_{c,k}\}$. The total test length required to test all $|C|$ cores on $w \geq |M|$ TAM wires is the maximum of the testing length on any of the CTAM chains. Let y_{cj} be a binary variable such that

$$y_{cj} = \begin{cases} 1, & \text{if a test for core } c \text{ involves CTAM chain } j, \\ 0, & \text{otherwise.} \end{cases}$$

Core c involves CTAM j , if $si_{c,j} + so_{c,j} + sl_{c,j} \neq 0$. Now, the total testing length of all child cores can be expressed as

$$T(w) = \max_{j=1}^{|M|} \sum_{c \in C} t_c \cdot y_{cj}. \quad (2)$$

Case 2: $w < |M|$.

If the number of available TAM wires w is less than the number of CTAM chains $|M|$, then the available TAM wires have to be distributed among the CTAM chains, such that the overall test length of the child cores is minimized. Two or more CTAM chains can be daisy-chained to form TAM chains that share the same TAM wire. However, the scan lengths of the cores are subject to change depending on the daisy chaining of the CTAM chains.

The scan-in and scan-out times of a core can now be defined as follows: Let two CTAM chains τ_1 and τ_2 be daisy-chained to form a TAM chain τ^* . Let C_{τ_1} and C_{τ_2} be the set of cores on CTAM terminal τ_1 and τ_2 , respectively. Assuming that τ_1 precedes τ_2 , the scan-in and scan-out times for a core c on τ_2 can now be defined for TAM chain τ^* as follows:

$$\begin{aligned} si_{c,\tau^*} &= \sum_{j \in C_{\tau_1}} sl_{j,\tau_1} + si_{c,\tau_2}, \\ so_{c,\tau^*} &= so_{c,\tau_2}. \end{aligned} \quad (3)$$

However, if τ_2 was to precede τ_1 , the scan-in and scan-out times for core c can be defined as follows:

$$\begin{aligned} si_{c,\tau^*} &= si_{c,\tau_2}, \\ so_{c,\tau^*} &= \sum_{j \in C_{\tau_2}} sl_{j,\tau_2} + so_{c,\tau_1}. \end{aligned} \quad (4)$$

It should be noted that a core can have its TAM items connected to both CTAM chains τ_1 and τ_2 , in which case the maximum of scan-in and scan-out times obtained from the above expressions is chosen as the scan-in and scan-out times of the core on τ^* .

The scan-in and scan-out times of every core are calculated as the maximum of the scan-in and scan-out times on every TAM chain, respectively. The number of TAM chains formed by daisy-chaining CTAM chains is equal to the number of available TAM wires w . Hence, the scan-in and scan-out times of the cores can be determined by taking the maximum of their scan times on the w TAM chains. The scan-in and scan-out times of a core c can be written as follows:

$$\begin{aligned} si_c &= \max_{1 \leq j \leq w} \{si_{c,j}\}, \\ so_c &= \max_{1 \leq j \leq w} \{so_{c,j}\}. \end{aligned} \quad (5)$$

Based on the updated scan-in and scan-out times as described above, the overall test length for a core c can be calculated by using (1). The overall test length for testing the child cores in INTEST_C mode can be determined using (2).

6.3 Distribution of TAM Wires

The distribution of available w TAM wires over the M CTAM chains should be such that the overall test length for child cores in INTEST_C mode is minimized. For this, a heuristic procedure COREWRAP(w, M) is described in Algorithm 1. The procedure consists of a short initialization, followed by two main steps. In the initialization step, we sort CTAM chains in the nonincreasing order of their test length. The test length of a TAM chain is equal to the sum of the test length of all cores connected to the TAM chain.

Algorithm COREWRAP(w, M)

```

1  sort CTAM chains  $\tau \in M$  such that
    $t(\{\tau_1\}) \leq \dots \leq t(\{\tau_{|M|}\})$ ; // sort CTAM chains in
   nonincreasing order of
   their test time
2   $\Gamma := \emptyset$ ; // initially there are no
   TAM sets
3  for  $i := 1$  to  $|M|$  // Step 1: create TAM sets
   with individuals CTAMs
4     $\gamma_i := \tau_i$ ; // create a TAM set and
   assign a CTAM to it
5     $\Gamma := \Gamma \cup \{\gamma_i\}$ ; // add the created TAM
   set to the set of TAM
   sets  $\Gamma$ 
6  for  $i := w + 1$  to  $|M|$  // Step 2: Concatenate
   extra TAM sets
7    find  $\gamma_{\min}$  for which  $t(\gamma_{\min}) = \min_{\gamma \in \Gamma} t(\gamma)$ ;
   // find the TAM set  $\gamma_{\min}$ 
   with the minimum
   test time
8     $t(\gamma^*) := \sum_{\gamma \in \Gamma} t(\gamma)$ ;
9    for all  $\gamma \in \Gamma \setminus \{\gamma_{\min}\}$  // for all TAM sets
10      $\gamma_{\text{temp}} := \gamma \cup \gamma_{\min}$ ; // concatenate a TAM set
   with  $\gamma_{\min}$ 
11     if  $t(\gamma_{\text{temp}}) < t(\gamma^*)$  // if concatenated TAM set
   has minimum test time
12        $\gamma^* := \gamma_{\text{temp}}$ ;  $\gamma_{\text{del}} := \gamma$ ;
   // accept this move
13    $\Gamma := \Gamma \setminus \{\gamma_{\text{del}}, \gamma_{\min}\} \cup \{\gamma^*\}$ ;
   // update the set of
   TAM Sets
14 return  $\Gamma$ ; // return the set of TAM
   sets  $\Gamma$ 

```

In Step 1 (lines 3-5), we assign each individual CTAM to a TAM set. The created TAM set is added to the set of TAM sets Γ . As $w < |M|$, we have $|M| - w$ TAM sets extra in Γ and these TAM sets need to be concatenated with other TAM sets. In Step 2 (lines 6-14), we concatenate TAM sets such that the resulting TAM sets have the overall minimum test length. In line 7, we find a TAM set with minimum test length. Next, we concatenate this TAM set with another TAM set that results in the minimum overall test length. Step 2 continues until $|M| - w$ CTAM chains have been daisy-chained with other TAM chains. The procedure COREWRAP returns a set of TAMs.

Wrapper-wide bypasses in child cores can help reduce test length in the INTEST_C mode. If wrapper-wide bypasses are present, the daisy chaining of CTAM chains does not increase the scan-in and scan-out times of the cores. In this case, when a child core is being tested on a particular TAM chain, the wrapper bypasses of all other cores on that TAM chain can be activated. As a result, the test stimuli for the core under test do not have to be scanned through other cores; this can minimize the test length of the core. However, wrapper bypasses cannot be activated during INTEST_P mode, since the wrapper cells of all cores have to participate in the testing of the parent core. In the INTEST_P mode, it is advantageous for the child cores to have scan chain bypasses, since this will minimize the scan lengths of the cores.

6.4 Test Architecture Design

Let us first consider the design scenario in which the wrapper and TAM architecture for all child cores in an SoC are fixed and given. Using the classification of Section 4, this scenario corresponds to $P(s, s, h)$; $C(h, h, h)$. From the test architecture design point of view, a test architecture design algorithm now needs to partition the total available TAM width over an optimal number of TAMs and assign only parent and other SoC-level nonhierarchical cores to these TAMs. To design the wrapper around a parent core for a given TAM width, the wrapper design strategy presented above should be used, while for the wrapper design for nonhierarchical cores, wrapper design procedure as described in [7] can be used. Therefore, integration of this solution into any test architecture design algorithm only requires changes in the wrapper design routine.

Now, let us consider another design scenario in which the wrapper and TAM architecture for all cores irrespective of being child or parent are soft. This scenario corresponds to $P(s, s, h)$; $C(s, s, h)$. In this case, the system integrator can design the wrapper and TAM architecture of the child cores in accordance with the TAM width available at the parent core interface. Therefore, concatenation of TAM wires connected to the child cores is not required. To design the test architecture in this case, we propose the following strategy. First, the SoC specification is preprocessed, such that only the parent and other SoC-level nonhierarchical cores are available to the TAM partitioning and core-assignment procedure. For a given TAM width, the test length for a parent core is calculated by considering the parent core as a small stand-alone SoC. The TAM partitioning and core-assignment procedure is used again to calculate the wrapper and TAM architecture for the child cores in the parent core. Based on the architecture thus obtained, the test length is computed as the sum of the parent core test length in the INTEST_P mode and the overall test length of the child core test architecture.

7 EXPERIMENTAL RESULTS

In this section, we present the experimental results for four SoCs taken from the ITC'02 SoC Test Benchmarks [20], namely p22810, p34392, p93791, and a586710. These four SoCs were selected because they are the only ones in the benchmark set with multiple levels of design hierarchy. We have only considered core-internal tests for all cores in the SoCs and tests for the top-level SoC are excluded from the discussion. In our experiments, we have modified and used TR-ARCHITECT [15], [16] for test architecture design as we have access to this tool. However, our proposed solutions are not limited to this design method only and any test architecture design algorithm can be used instead. First, we compare the test length results for four cases.

Case 1 is the original test architecture design as presented in [15]. In this case, all cores in an SoC are considered to be at the same level of design hierarchy. Therefore, in all SoCs, all cores are considered to be *flat*. It is important to note that in Case 1, since no hierarchy is assumed, testing of a core requires access to its own scan chains and wrapper cells only.

In Case 2, we assume that the test architecture for an SoC is already designed as in Case 1 and we are only allowed to modify the test schedule in order to respect the hierarchy present in the SoC. In this case, the wrapper of a child core is

TABLE 2
Test Length Results for the Four Cases

SOC	w	Flat Cores	Hierarchical Cores					
		[15] (Case 1)	Modified Schedule (Case 2)		Modified Wrapper Cell (Case 3)		New Wrapper Architecture (Case 4)	
		T [15]	T	ΔT	T	ΔT	T	ΔT
p22810	16	458068	621895	36%	466667	2%	530778	16%
	24	299718	547039	83%	309641	3%	343942	15%
	32	222471	493290	122%	229899	3%	288273	30%
	40	190995	460885	141%	191978	1%	263624	38%
	48	160221	279949	75%	157226	-2%	251299	57%
	56	145417	315469	117%	145417	0%	238974	64%
	64	133405	296445	122%	133405	0%	238974	79%
p34392	16	1010821	2327127	130%	1019766	1%	1154719	14%
	24	680411	1977631	191%	702852	3%	774221	14%
	32	551778	1449581	163%	584524	6%	606261	10%
	40	544579	643939	18%	544579	0%	593924	9%
	48	544579	855644	57%	544579	0%	581588	7%
	56	544579	855644	57%	544579	0%	581588	7%
	64	544579	855644	57%	544579	0%	581588	7%
p93791	16	1791638	3363819	88%	1792354	0%	1854566	4%
	24	1185434	3438009	190%	1211510	2%	1272220	7%
	32	912233	1504806	65%	917246	1%	940318	3%
	40	718005	4021279	460%	730713	2%	765715	7%
	48	601450	1965642	227%	610037	1%	640488	6%
	56	528925	1059869	100%	528407	0%	551849	4%
	64	455738	1908099	319%	458600	1%	473726	4%
a586710	16	41523868	60294453	45%	41523868	0%	42117536	1%
	24	28716501	53295859	86%	28716501	0%	28716501	0%
	32	22475033	40729137	81%	22475033	0%	21058768	-6%
	40	19048835	36478145	91%	19048835	0%	19144334	1%
	48	15315467	21723090	42%	15212440	-1%	15315467	0%
	56	13401034	13401034	0%	13401034	0%	13401034	0%
	64	12700205	12769440	1%	12510356	-1%	12510356	-1%

configured in EXTEST mode during the test of its parent core. Here, testing of a hierarchical core not only requires access to its own scan chains and wrapper cells but also to the wrapper cells of its child cores. The conventional wrapper cell is used in the wrapper architecture for all cores.

In Cases 3 and 4, we take into account the design hierarchy in the SoCs from the outset. In Case 3, we use the wrapper cells proposed in Section 5 for all child cores. Here, we modify the wrapper design algorithm used in TR-ARCHITECT. In Case 4, we consider the new wrapper architecture proposed in Section 6 and modify TR-ARCHITECT accordingly. The experimental setup for Case 4 is the same as that for Case 3, which requires wrappers and TAMs for all cores to be soft. This is done in order to compare the results of Case 3 (MWC) and Case 4 (new wrapper architecture). Using the three-tuple classification in Section 4, all four cases corresponds to a scenario with $P(s, s, h)$; $C(s, s, h)$. Note that the scan chains in the parent and child cores are assumed to be hard for all cores.

Table 2 shows the test length (in clock cycles) results for the four above-mentioned cases. The first two columns in Table 2 show the SoC name and the number of TAM wires $w_{\max}$ available for the SoC test architecture design. Column 3 shows the test length results [15] for Case 1, i.e., for flat cores. Column 4 shows the test length results for Case 2, in which we modify the test schedules obtained from Case 1 in order to respect design hierarchy. Column 5 shows the difference ($\Delta T = \frac{T(\text{Case2}) - T(\text{Case1})}{T(\text{Case1})}$) between the test lengths for Case 1 and Case 2. Columns 6 and 7 show the test length results for Case 3, while Columns 8 and 9 show the results for Case 4.

From Table 2, we make the following observations. If the design hierarchy is taken into account and only the test schedule is modified (Case 2), an average increase of 113 percent in test length is obtained compared to Case 1 (flat cores). For SoC p93791 with $w = 40$, the penalty in test length is more than 400 percent. This increase can be attributed to the fact that the design hierarchy is considered only as an afterthought. Hence, an approach that can handle hierarchy efficiently is needed for testing of SoCs.

The test length results for Case 3 show that with the use of the proposed wrapper cells in the child core wrapper, hierarchy-aware test length can be comparable with or even better than the test length obtained in a hierarchy-oblivious manner. From Column 6, we can see that for most cases, we obtain the same test length as for the SoCs with flat cores. It is important to note here that for Case 3, we obtain lower test length for some cases. This is due to the fact that the test architecture algorithm (TR-ARCHITECT) used here is heuristic in nature. As TR-ARCHITECT considers access to both the parent and child cores for hierarchical cores during the test architecture optimization itself, it results in a new TAM assignment. Therefore, due to different core assignments and TAM partitions, we sometimes obtain lower test length.

For SoC p93791 with $w = 40$, Fig. 15 shows the test schedules for Cases 1, 2, and 3. The horizontal axis represents the test length, while the vertical axis represents the TAM width. The number inside the box represents the core identification. Note that the schedules shown are not drawn to scale, and for the sake of clarity, idle time is shown at the end of a TAM only. Fig. 15a shows an effective test schedule if hierarchy is not considered. However, the problem of an unbalanced test schedule is shown in Fig. 15b. The dark gray boxes show the tests for

Fig. 15. For SoC p93791 with $w = 40$, test schedules for three cases. (a) Case 1. (b) Case 2. (c) Case 3.

the child cores in their EXTEST mode. The black boxes denote idle times in the test schedule. We can see that for TAM 5 (of width 5 bits; the second TAM partition from the bottom), which contains a large number of child cores, the EXTEST mode for the child cores dominates. Therefore, this TAM determines the overall test length.

The use of proposed wrapper cells in the child core wrappers results in the test schedule shown in Fig. 15c. This shows that just by using 1,875 new wrapper cells, test length close to the one with flat cores can be obtained.

The test length results for Case 4 show a slight increase over the test length results obtained for Case 3. This is expected since Case 3 allows parallel testing of parent and child cores. Such parallel testing is not possible in Case 4 due to time-multiplexing of the parent and child core INTEST modes. However, for SoC a586710 with $w = 32$, the use of new wrapper architecture resulted in 6 percent reduction in test length as compared to the test length with all flat cores. Also, the test length for Case 4 are far less than that for Case 2. From the results, we conclude that the use of proposed wrapper cells in the child core's wrapper results in the minimum test length.

Next, we compare the area cost for Case 3 and Case 4. In Case 2, no additional area is required as we only modify the test schedule. The test architecture or the wrapper design is not affected. Table 3 shows the area costs for Case 3 and Case 4. The area costs are quantified in terms of equivalent number of additional NAND2 gates. It is assumed that a 2-to-1 multiplexer is equivalent to three NAND2 gates in terms of area, while a flip-flop is equivalent to seven NAND2 gates. The first two columns in Table 3 list the SoC name and the number of TAM wires w available for the SoC test architecture design. Column 3 shows the number of MWCs required in Case 3, while Column 4 shows the same information but in terms of equivalent number of NAND2 gates. Column 5 shows the equivalent number of NAND2 gates required for the new wrapper architecture used in Case 4. Column 6 highlights the difference ($\Delta A = \frac{NAND2Eq.(Case4) - NAND2Eq.(Case3)}{NAND2Eq.(Case3)}$) between the equivalent number of NAND2 gates for Case 3 and Case 4.

It is clear from the table that the area cost for the new wrapper architecture is lower than that for the design based on the MWC. The area cost for the approach with MWCs depends on the total number of child cores in an SoC and the number of the functional terminals in them. We therefore conclude from Tables 2 and 3 that the use of MWCs results in lower test length, while the use of new wrapper architecture results in lower area cost.

The values of MWC can change depending on the problem instance. This can be explained as follows: In the worst case, all child cores in the SoC can be equipped with the new wrapper cell. In that case, MWC will be constant for an SoC and will not change with w . However, this is not really required as only the child cores connected to TAMs that are

TABLE 3
Comparison of Area-Cost Results for Case 3 and Case 4

SOC	w	Modified Wrapper Cell (Case 3)		New Wrapper Architecture (Case 4)	
		MWC	NAND2 Eq.	NAND2 Eq.	ΔA
p22810	16	600	61481	55721	-9.36%
	24	600	61481	55730	-9.35%
	32	600	61481	55736	-9.34%
	40	600	61481	55739	-9.33%
	48	600	61481	55760	-9.30%
	56	548	61021	55790	-8.57%
p34292	64	600	61481	55817	-9.21%
	16	624	32435	26789	-17.40%
	24	336	29540	26813	-9.23%
	32	624	32435	26834	-17.26%
	40	301	29478	26852	-8.90%
	48	388	30237	26879	-11.10%
p93791	56	388	30237	26900	-11.03%
	64	388	30237	26900	-11.03%
	16	886	95379	87179	-18.59%
	24	2157	106880	87323	-18.29%
	32	2007	105422	87311	-17.17%
	40	1875	104342	87359	-16.27%
a586710	48	2127	106496	87281	-18.04%
	56	2270	107875	87380	-18.99%
	64	2127	106496	87455	-17.87%
	16	1678	47764	31293	-34.48%
	24	1678	47764	31290	-34.49%
	32	1678	47764	31305	-34.45%
	40	1678	47764	31302	-34.46%
	48	1678	47764	31302	-34.46%
	56	1148	41830	31305	-25.16%
	64	1148	41830	31308	-25.15%

different than their parent TAM need to have this new wrapper cell. If parent and child cores share the same TAM, then the child cores can be configured in the EXTEST mode the parent core is tested. As for different w values, note that we have different core assignments for the various cases; therefore, different values of MWC result.

8 CONCLUSION

SoC designs today are no longer limited to only one level of hierarchy (SoC and cores). Instead, they typically consist of multiple levels of design hierarchy. In this paper, we have addressed the problem of efficient testing of SoCs with hierarchical cores. We have shown that the existing wrapper architectures including the IEEE Std. 1500 and test architecture design algorithms (originally developed for SoCs with flat cores) are not directly applicable to the testing of SoCs with hierarchical cores. We have presented two hierarchy-aware approaches for the testing of hierarchical cores. The two proposed solutions can be used in conjunction with any test architecture design algorithm and they require very little modification in the architecture optimization procedure.

We have presented experimental results for four hierarchical SoCs taken from the ITC'02 SoC Test Benchmarks. We have shown that if the design hierarchy is taken into account, no modification is allowed in the test architecture, and only the test schedule is modified, up to 400 percent penalty in test length is incurred. The use of the proposed wrapper cells in the child core wrapper allows us to obtain near-optimal test length even when design hierarchy is taken into account. The use of the new wrapper architecture results in slightly higher test length as compared to the use of the MWC, but it offers the advantage of lower area cost.

ACKNOWLEDGMENTS

Parts of this paper were published in the *Proceedings of the IEEE International Test Conference 2004* and the *Proceedings of the Design, Automation and Test in Europe (DATE) Conference 2006*. Sandeep Kumar Goel and Erik Jan Marinissen were at Philips Research at the time this research work was carried out. Their work was supported in part by the European MEDEA + Program (Project 2A702 NanoTEST). Anuja Sehgal and Krishnendu Chakrabarty were at Duke University at the time this research work was carried out. Their work was supported in part by the US National Science Foundation under Grant CCR-9875324 and Grant CCR-0204077 and in part by the Semiconductor Research Corporation under Contract 2004-TJ-1174.

REFERENCES

- [1] S. Dutta, R. Jensen, and A. Rieckmann, "VIPER: A Multiprocessor SoC for Advanced Set-Top Box and Digital TV Systems," *IEEE Design and Test of Computers*, pp. 21-31, 2001.
- [2] S.K. Goel, K. Chiu, E.J. Marinissen, T. Nguyen, and S. Oostdijk, "Test Infrastructure Design for the Nexperia Home Platform PNX8550 System Chip," *Proc. Design, Automation, and Test in Europe (DATE '04) Designers Forum*, pp. 108-113, Feb. 2004.
- [3] Y. Zorian, E.J. Marinissen, and S. Dey, "Testing Embedded-Core-Based System Chips," *Computer*, vol. 32, no. 6, pp. 52-60, June 1999.
- [4] Y. Zorian, E.J. Marinissen, and S. Dey, "Testing Embedded-Core Based System Chips," *Proc. IEEE Int'l Test Conf. (ITC '98)*, pp. 130-143, Oct. 1998.
- [5] E.J. Marinissen et al., "A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores," *Proc. IEEE Int'l Test Conf. (ITC '98)*, pp. 284-293, Oct. 1998.
- [6] P. Varma and S. Bhatia, "A Structured Test Re-Use Methodology for Core-Based System Chips," *Proc. IEEE Int'l Test Conf. (ITC '98)*, pp. 294-302, Oct. 1998.
- [7] E.J. Marinissen, S.K. Goel, and M. Lousberg, "Wrapper Design for Embedded Core Test," *Proc. IEEE Int'l Test Conf. (ITC '00)*, pp. 911-920, Oct. 2000.
- [8] IEEE Std. 1500-2005, *IEEE Standard Testability Method for Embedded Core-Based Integrated Circuits*, F. DaSilva, ed., IEEE, Aug. 2005.
- [9] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip," *Proc. IEEE Int'l Test Conf. (ITC '01)*, pp. 1023-1032, Oct. 2001.
- [10] S. Koranne, "Design of Reconfigurable Core Wrappers for Embedded Core Based SoC Test," *Proc. Third Int'l Symp. Quality of Electronic Design (ISQED '02)*, Mar. 2002.
- [11] K. Chakrabarty, "Optimal Test Access Architectures for System-on-a-Chip," *ACM Trans. Design Automation of Electronic Systems*, vol. 6, no. 1, pp. 26-49, Jan. 2001.
- [12] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "Co-Optimization of Test Wrapper and Test Access Architecture for Embedded Cores," *J. Electronic Testing: Theory and Applications*, vol. 18, no. 2, pp. 213-230, Apr. 2002.
- [13] Z. sadat-Ebadi and A. Ivanov, "Design of an Optimal Test Access Architecture Using a Genetic Algorithm," *Proc. 10th IEEE Asian Test Symp. (ATS '01)*, pp. 205-210, Nov. 2001.
- [14] Y. Huang et al., "Resource Allocation and Test Scheduling for Concurrent Test of Core-Based SoC Design," *Proc. 10th IEEE Asian Test Symp. (ATS '01)*, pp. 265-270, Nov. 2001.
- [15] S.K. Goel and E.J. Marinissen, "Effective and Efficient Test Architecture Design for SoCs," *Proc. IEEE Int'l Test Conf. (ITC '02)*, pp. 529-538, Oct. 2002.
- [16] S.K. Goel and E.J. Marinissen, "SoC Test Architecture Design for Efficient Utilization of Test Bandwidth," *ACM Trans. Design Automation of Electronic Systems*, vol. 8, no. 4, pp. 399-429, Oct. 2003.
- [17] Y. Huang et al., "Optimal Core Wrapper Width Selection and SoC Test Scheduling Based on 3-D Bin Packing Algorithm," *Proc. IEEE Int'l Test Conf. (ITC '02)*, pp. 74-82, Oct. 2002.
- [18] S. Koranne and V. Iyengar, "On the Use of k -Tuples for SoC Test Schedule Representation," *Proc. IEEE Int'l Test Conf. (ITC '02)*, pp. 539-548, Oct. 2002.
- [19] E. Larsson and H. Fujiwara, "Test Resource Partitioning and Optimization for SoC Designs," *Proc. 21st IEEE VLSI Test Symp. (VTS '03)*, pp. 319-324, Apr. 2003.
- [20] E.J. Marinissen, V. Iyengar, and K. Chakrabarty, "A Set of Benchmarks for Modular Testing of SoCs," *Proc. IEEE Int'l Test Conf. (ITC '02)*, pp. 519-528, Oct. 2002.
- [21] M. Benabdenbi, W. Maroufi, and M. Marzouki, "CAS-BUS: A Scalable and Reconfigurable Test Access Mechanism for Systems on a Chip," *Proc. Design, Automation, and Test in Europe (DATE '00)*, pp. 141-145, Mar. 2000.
- [22] K.-J. Lee and C.-I. Huang, "A Hierarchical Test Control Architecture for Core-Based Design," *Proc. Ninth IEEE Asian Test Symp. (ATS '00)*, pp. 248-253, 2000.
- [23] J.-F. Li, H.-J. Huang, J.-B. Chen, C.-P. Su, C.-W. Wu, C. Cheng, S.-I. Chen, C.-Y. Hwang, and H.-P. Lin, "A Hierarchical Test Scheme for System-on-Chip Designs," *Proc. Design, Automation, and Test in Europe (DATE '02)*, pp. 486-490, 2002.
- [24] Q. Xu and N. Nicolici, "Resource-Constrained System-on-a-Chip Test: A Survey," *IEE Proc., Computers and Digital Techniques*, vol. 152, nos. 4/5, pp. 67-81, Jan. 2005.
- [25] D. Zhao and S. Upadhyaya, "Dynamically Partitioned Test Scheduling with Adaptive TAM Configuration for Power-Constrained SoC Testing," *IEEE Trans. Computer-Aided Design*, vol. 24, no. 6, pp. 956-965, June 2005.
- [26] E. Larsson and Z. Peng, "Power-Aware Test Planning in the Early System-On-Chip Design Exploration Process," *IEEE Trans. Computers*, vol. 6, no. 2, pp. 227-239, Feb. 1996.
- [27] T. Yoneda, M. Imanishi, and H. Fujiwara, "An SoC Test Scheduling Algorithm Using Reconfigurable Union Wrappers," *Proc. Design, Automation, and Test in Europe (DATE '07)*, pp. 231-236, Apr. 2007.
- [28] V. Iyengar, K. Chakrabarty, M.D. Krasniewski, and G.N. Kumar, "Design and Optimization of Multi-Level TAM Architectures for Hierarchical SoCs," *Proc. 21st IEEE VLSI Test Symp. (VTS '03)*, pp. 299-304, Apr. 2003.

- [29] Q. Xu and N. Nicolici, "Time/Area Tradeoffs in Testing Hierarchical SoCs with Hard Mega-Cores," *Proc. IEEE Int'l Test Conf. (ITC '04)*, pp. 1196-1202, Oct. 2004.
- [30] J. Pouget, E. Larsson, and Z. Peng, "Multiple-Constraint Driven System-on-Chip Test Time Optimization," *J. Electronic Testing: Theory and Applications*, vol. 21, pp. 599-611, 2005.
- [31] T. Waayers, R. Morren, and R. Grandi, "Definition of a Robust Modular SoC Test Architecture; Resurrection of the Single TAM Daisy-Chain," *Proc. IEEE Int'l Test Conf. (ITC '05)*, Nov. 2005.
- [32] S.K. Goel, "Test-Access Planning and Test Scheduling for Core-Based System Chips," PhD thesis, Univ. of Twente, Feb. 2005.
- [33] S.K. Goel, *Test Circuit and Method for Hierarchical Core*, US Patent 7380181, May 2008.
- [34] F. DaSilva, Y. Zorian, L. Whetsel, K. Arabi, and R. Kapur, "Overview of the IEEE P1500 Standard," *Proc. IEEE Int'l Test Conf. (ITC '03)*, pp. 988-997, Sept. 2003.
- [35] A. Sehgal, "Test Infrastructure Design for Digital, Mixed-Signal and Hierarchical SoCs," PhD thesis, Duke Univ., May 2005.
- [36] E.J. Marinissen et al., "On IEEE P1500's Standard for Embedded Core Test," *J. Electronic Testing: Theory and Applications*, vol. 18, no. 4/5, pp. 365-383, Aug. 2002.
- [37] E.J. Marinissen, "The Role of Test Protocols in Automated Test Generation for Embedded-Core-Based System ICs," *J. Electronic Testing: Theory and Applications*, vol. 18, nos. 4/5, pp. 435-454, Aug. 2002.

Sandeep Kumar Goel received the BTech degree from the Institute of Engineering and Technology, Lucknow, in 1998, the MTech degree from the Indian Institute of Technology, Delhi, in 1999, and the PhD degree from the University of Twente, Enschede, Netherlands, in 2005. He is currently a principal engineer at LSI, Milpitas, California. Prior to LSI, he was with Magma Design Automation, San Jose and Philips Research, Eindhoven, Netherlands. His

research interests include ATPG, BIST, test compression, test cost reduction techniques, debug and diagnosis, SoC test, Yield improvement, and other DFM-related issues. He is a coauthor of more than 35 journal and conference papers. He is a coinventor of two granted US patents and has four other patents pending. He serves on various conference and workshop committees, including Design, Automation and Test in Europe (DATE), IEEE Infrastructure IP Workshop (IIP), IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS), and IEEE International Workshop on Defect and Data Driven Testing (D3T). He is a senior member of the IEEE.

Erik Jan Marinissen received the MSc degree in computing science and the PDEng degree in software technology from Eindhoven University of Technology in 1990 and 1992, respectively. He is currently a principal scientist at IMEC, Leuven, Belgium. Prior to IMEC, he was with NXP Semiconductors and Philips Research, both Eindhoven, The Netherlands. His research interests include all topics in the domain of test and debug of digital VLSI circuits. He is a coauthor of

more than 100 journal and conference papers and a coinventor on seven granted US patents. He is a recipient of Best Paper Awards at the Chrysler-Delco-Ford Automotive Electronics Reliability Workshop in 1995, the IEEE International Board Test Workshop in 2002, and the Most Significant Paper Award at the IEEE International Test Conference in 2008. He served as an editor in chief of IEEE Std. 1500. He serves on numerous conference committees, including Asian Test Symposium (ATS), European Test Symposium (ETS), Design, Automation and Test in Europe (DATE), International Test Conference (ITC), and IEEE VLSI Test Symposium (VTS), and on the editorial boards of *IEEE Design and Test of Computers*, *IET Computers and Digital Techniques (IET-CDT)*, and Springer's *Journal of Electronic Testing: Theory and Applications (JETTA)*. He is a senior member of the IEEE and a golden core member of Computer Society.

She is a member of the IEEE.

Anuja Sehgal received the BE degree in electronics engineering from the Ramrao Adik Institute of Technology, University of Mumbai, Nerul, India, in 2001 and the MS and PhD degrees in electrical and computer engineering from Duke University in 2003 and 2005, respectively. She is currently a senior design engineer at NVIDIA, Santa Clara, California. Her research interests lie in test cost reduction techniques, MBIST, and ATPG. She is a recipient of the Best Speaker Award at the 2004 North Atlantic Test Workshop and a Best Paper Award at the 2005 IEEE International Conference on Computer Design.

of system-on-chip integrated circuits, digital microfluidic biochips, and nanotechnology circuits and systems based on DNA self-assembly. He has coauthored seven books, contributed 15 invited chapters to book volumes, published 300 papers in journals and refereed conference proceedings, and delivered more than 120 keynote, plenary, and invited talks. He holds a US patent in built-in self-test and is a coinventor of a pending US patent on sensor networks. He is a recipient of the National Science Foundation Early Faculty (CAREER) Award (1999), the Office of Naval Research Young Investigator Award (2001), and the Humboldt Research Fellowship, awarded by the Alexander von Humboldt Foundation, Bonn, Germany, in 2004. He is also a recipient of Best Paper Awards at the 2007 IEEE International Conference on VLSI Design, the 2005 IEEE International Conference on Computer Design, and the 2001 IEEE Design, Automation and Test in Europe (DATE) Conference. He served as a distinguished visitor of the IEEE Computer Society for 2005-2007 and as a distinguished lecturer of the IEEE Circuits and Systems Society for 2006-2007. Since 2008, he has been serving as an ACM distinguished speaker. He is an associate editor of *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *IEEE Transactions on VLSI Systems*, *IEEE Transactions on Biomedical Circuits and Systems*, and *ACM Journal on Emerging Technologies in Computing Systems*, and an editor of *IEEE Design and Test of Computers* and *Journal of Electronic Testing: Theory and Applications (JETTA)*. He is a recipient of Duke University's 2008 Dean's Award for Excellence in Mentoring and a Meritorious Service Award from the IEEE Computer Society in 2008. He served as a program chair for the IEEE Asian Test Symposium in 2005 and as the program vice-chair for the IEEE International Mixed-Signals, Sensors, and Systems Test Workshop in 2008. He is a fellow of the IEEE, a Distinguished Engineer of ACM, and a member of Sigma Xi and a 2009 Fellow of the Japan Society for the Promotion of Science.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.