DFT Infrastructure for Broadside Two-Pattern Test of Core-Based SOCs

Qiang Xu, *Member*, *IEEE*, and Nicola Nicolici, *Member*, *IEEE*

Abstract—Existing approaches for modular manufacturing test of core-based system-on-a-chip (SOC) devices do not provide any explicit mechanism for delivering two-pattern tests in the broadside mode, which is necessary to achieve reliable coverage of delay and stuck-open faults. Although wrapper input cells can be enhanced with two memory elements to address this problem, this will incur a large test area overhead. This paper proposes a novel architecture for broadside two-pattern test of core-based SOCs without any loss in fault coverage and without increasing the size of the wrapper input cells. The proposed solution combines the dedicated bus-based test access mechanism and functional interconnects for test data transfer in order to provide full controllability of the wrapper input cells in the two consecutive clock cycles required by two-pattern testing. New algorithms for test access mechanism design and test scheduling are proposed and design trade-offs between test area and testing time are discussed using experimental results.

Index Terms—System-on-a-chip, embedded core delay test.

1 Introduction

One way to model digital integrated circuits' (ICs) physical defects is to abstract them as stuck-at faults [4]. The stuck-at fault model has been extensively studied and was shown to be effective in verifying the logic correctness of digital circuits. However, digital ICs are generally synchronized using clock signals, which may also lead to timing failures without logic errors, i.e., circuits fail to operate at the specified speed but can produce correct outputs at a slower or faster speed. Although application of stuck-at fault tests at rated-speed can uncover some delay defects, it was shown in [6], [30] that this technique is not sufficient. Functional testing can be used to address timing verification and even detection of unmodeled defects; nevertheless, its main drawback lies in the low fault coverage for today's complex circuits, whose transistor to pin ratio is continuing to increase. Moreover, with the shrinking feature size of the very large scale integration (VLSI) technology, more timing-related defects are emerging [7]. As a result, to increase circuit reliability and manufacturing yield through speed sorting, semiconductor manufacturers are constrained to develop delay fault tests and the associated testing strategies.

To apply delay fault tests, at least two ordered patterns in *consecutive* clock cycles are necessary: The first *launch* (*initialization*) pattern V_1 initializes the circuit to a certain state and then the second *capture* (*excitation*) pattern V_2 provokes the fault and captures its effect on the outputs. In addition to testing delay faults, two-pattern tests can also be used to detect the CMOS stuck-open faults, which are used to model the defects that cause the transistors to be permanently off [34]. It should be noted, however, when applying two-pattern tests for delay faults, the outputs must be sampled close to its operating frequency, while, for stuck-open faults, they can be applied at slower speed.

There are three main approaches to extend the mainstream scan-based testing technique to handle two-pattern tests [33]: 1) *enhanced scan*, which requires two sequential elements in every scan cell, 2) *skewed-load testing* (also called launch-on-shift, launch-from-shift, or last-shift-launch), which uses the last-shifted pattern in the scan chain as the excitation vector, and 3) *broadside testing* (also called launch-on-capture, launch-from-capture, or functional justification), where the sequential (pseudo-input) part of the second pattern is generated through the combinational block. It is important to note that primary inputs (PIs) are assumed to be fully controllable in all three techniques.

Although enhanced scan can apply any arbitrary vector pairs and, hence, test generation is easier than the other two techniques, it is rarely used because of its large area and performance overhead [36]. Both skewed-load testing and broadside testing, however, work with standard scan design (each scan cell contains only one memory element) and are widely accepted in practice. On the one hand, skewed-load testing offers an advantage over broadside testing mainly in the ease of test generation and less test pattern count. However, its main drawback is that the scan enable (SE) signal must operate at rated speed because it must toggle before and after the capture edge of the high-speed clock [24], [25], [32]. The strict delay and skew design requirements for SE signal pose a big challenge for physical design of the circuit. On the other hand, the timing of SE in broadside testing is not critical because both launch and capture occur in normal functional mode. In addition, test application via skewed-load testing delivers tests that may not be sensitizable in the normal operation, which can cause unnecessary yield loss [33]. In contrast, broadside testing

- *Q. Xu is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong. E-mail: qxu@cse.cuhk.edu.hk.*
- *N. Nicolici is with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON, L8S 4K1, Canada. E-mail: nicola@ece.mcmaster.ca.*

Manuscript received 22 Dec. 2004; revised 12 July 2005; accepted 2 Nov. 2005; published online 22 Feb. 2006.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0424-1204.

0018-9340/06/$20.00 © 2006 IEEE Published by the IEEE Computer Society

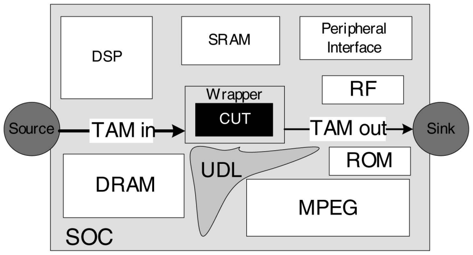


Fig. 1. Conceptual infrastructure for SOC testing [43].

limits the space of the possible consecutive patterns to only those that affect the timing in the normal mode, thus covering the worst-case operational behavior of the manufactured circuit. Therefore, in this paper, we focus on the broadside testing technique.

Since there has been extensive research on design for test (DFT) and automatic test pattern generation (ATPG) for two-pattern tests over the last couple of decades [4], the question is how do the existing methods adapt to core-based SOCs [43]? This adaptation is an open issue since, in addition to the standard test-quality problems, core-based SOCs present new challenges, in particular, in terms of test development for core providers and test access mechanism (TAM) design for system integrators. Furthermore, as pointed out in [22] by Kapur and Williams, *a higher test quality for each core is required to achieve acceptable overall quality of the SOC, when compared to the case that the core itself is a chip*.

Although IEEE Std. 1500 (i.e., the IEEE standard for embedded core test) [14], [29] and recent research advances [41] support structural scan tests, to the best of our knowledge, most prior work in this domain is based on one-pattern test. *In this paper, we propose a novel test architecture for two-pattern test of core-based SOCs with limited DFT area using the broadside testing approach, where test data is transferred through both the dedicated bus-based TAMs and functional interconnects between cores*. We also adapt an existing approach to optimize the new test architecture in terms of test application time (TAT). Experiments on the ISCAS89 [3] and ITC'02 [28] benchmark circuits are presented to show the benefits of the proposed methodology.

The rest of this paper is organized as follows: Section 2 reviews the related work on testing core-based SOC and Section 3 motivates the research presented in this paper. The proposed SOC test architecture for two-pattern test and the necessary core-level DFT support are presented in Section 4. Next, in Section 5, we adapt an existing wrapper/TAM cooptimization algorithm to optimize the proposed architecture at the system level. Section 6 contains our experimental results. Finally, Section 7 concludes this paper.

2 Prior Work

Zorian et al. [43] proposed a conceptual SOC test architecture (as illustrated in Fig. 1) that employs a modular test approach in which special TAMs are built to transport test data between test sources/sinks (e.g., tester) and the core under test (CUT). Embedded cores are isolated from surrounding logic and connected to the TAMs using core wrappers. This "divide and conquer" test strategy helps to reduce the test generation time and facilitates test reuse.

2.1 Test Access to Embedded Cores

One of the major challenges for SOC testing is to design an efficient TAM to link the test sources and sinks to the CUT. There are a number of solutions for accessing the embedded cores from chip's I/O pins [43]:

1. direct parallel access via pin muxing,
2. serial access and core isolation through a boundary scan-like architecture (also called isolation ring access mechanism),
3. functional access through functional buses or transparency of embedded cores, and
4. access through a combination of core wrappers and dedicated test buses.

Direct access strategy [15] introduces a large routing overhead and does not scale well when the number of embedded cores and/or core terminals is large. Isolation ring access [35], [39] solved the above problems, however, at the cost of very long test application time. Functional access [5], [8], [31], [42] significantly reduce the DFT hardware cost by introducing core transparency or using the existing functional paths as test paths. These test strategies, however, dramatically increase the system integrator's test planning effort, especially for complex SOCs with a large number of cores and, hence, are not widely accepted in industry.

Since time-to-market is the overriding goal of core-based design, the ease with which cores can be designed and tested is crucial. Therefore, the increased size of the logic and routing resources consumed by dedicated test infrastructure is acceptable for large SOC designs. Aerts and Marinissen [1] described three basic types of bus-based test architectures: 1) the *Multiplexing* architecture, 2) the *Daisychain* architecture, and 3) the *Distribution* architecture. In the *Multiplexing* and the *Daisychain* architecture, all cores get full access to all TAMs, while, in the *Distribution* architecture, the total TAM wires are distributed over all cores. Two more popular architectures that support more flexible test schedules are proposed based on the above architectures: The *Test Bus* architecture presented in [37] can be seen as a combination of the multiplexing and the distribution architectures, while the *TestRail* architecture proposed in [26] is a combination of the daisychain and the distribution architecture.

Most of the relevant research on SOC testing, as shown in the following sections, has been focused on dedicated test bus access, where the embedded cores are connected to the TAM wires using core wrappers, due to its modularity, scalability, and flexibility.

2.2 Embedded Core Wrapper Design and Optimization

Marinissen et al. [26] described a scalable core wrapper called *TestShell*, which forms the basis for the IEEE Std. 1500 core wrapper [14]. *Wrapper boundary register* (WBR) cell is used in the wrapper to provide controllability and observability for each core terminal. In the INTEST mode,

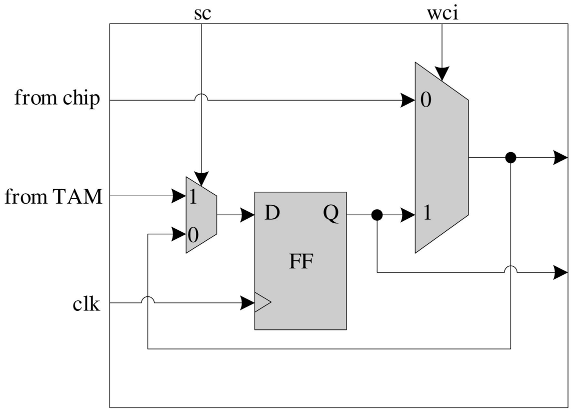


Fig. 2. A regular wrapper input register cell implementation [29].

used for testing the core's internal logic, the input WBR cells act as primary inputs to the CUT, while the output WBR cells act as primary outputs. In the EXTEST mode, when all the embedded cores are wrapped, the goal is to test the interconnect wires or logic between the cores. Thus, the output WBR cells provide stimuli and the input WBR cells capture responses from the interconnect, which are blocked in the input WBR cells and do not get propagated to the core's internal logic. A typical wrapper input cell (WIC) implementation is shown in Fig. 2, which contains only one flip-flop and, hence is not able to provide two-pattern tests in consecutive clock cycles. It is important to note, however, that IEEE Std. 1500 standardizes only the wrapper interface and, hence, the internal structure of the wrapper can be adapted to the specific SOC test requirements.

To the best of our knowledge, Vermaak and Kerkhoff [38] presented the only 1500-compatible wrapper design for delay fault testing in the literature, based on the digital oscillation test method introduced in [2]. To be able to use this method, they introduced extra multiplexers and a cell address register to each WBR cell. Consequently, this method is expensive in terms of DFT area. In addition, this approach is only suitable for combinational cores because only paths between the core's inputs and outputs are tested.

Since the TAT of a core is dependent on the maximum wrapper SC length, the main objective in wrapper optimization is to build balanced wrapper scan chains (wrapper SCs). Marinissen et al. [27] addressed this problem by describing a $COMBINE$ heuristic for hard cores. Later, Iyengar et al. [17] proposed the $Design\_wrapper$ algorithm based on the Best Fit Decreasing heuristic for the Bin Packing problem, which tries to minimize the core's TAT and required TAM width at the same time. They also showed an important feature of wrapper optimization for hard cores, i.e., the TAT varies with TAM width as a "staircase" function. According to this feature, only a few TAM widths between 1 and W_{ttl} (the total TAM width) are relevant when assigning TAM resources to hard cores and these discrete widths are called *pareto-optimal* TAM widths.

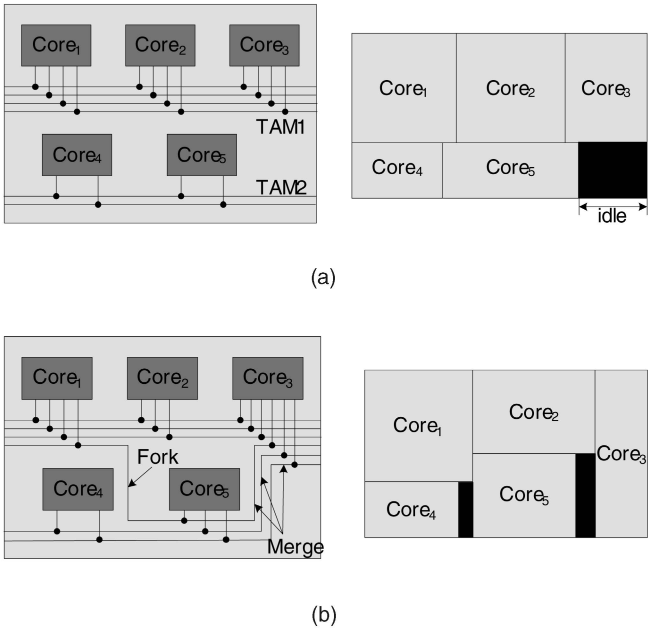


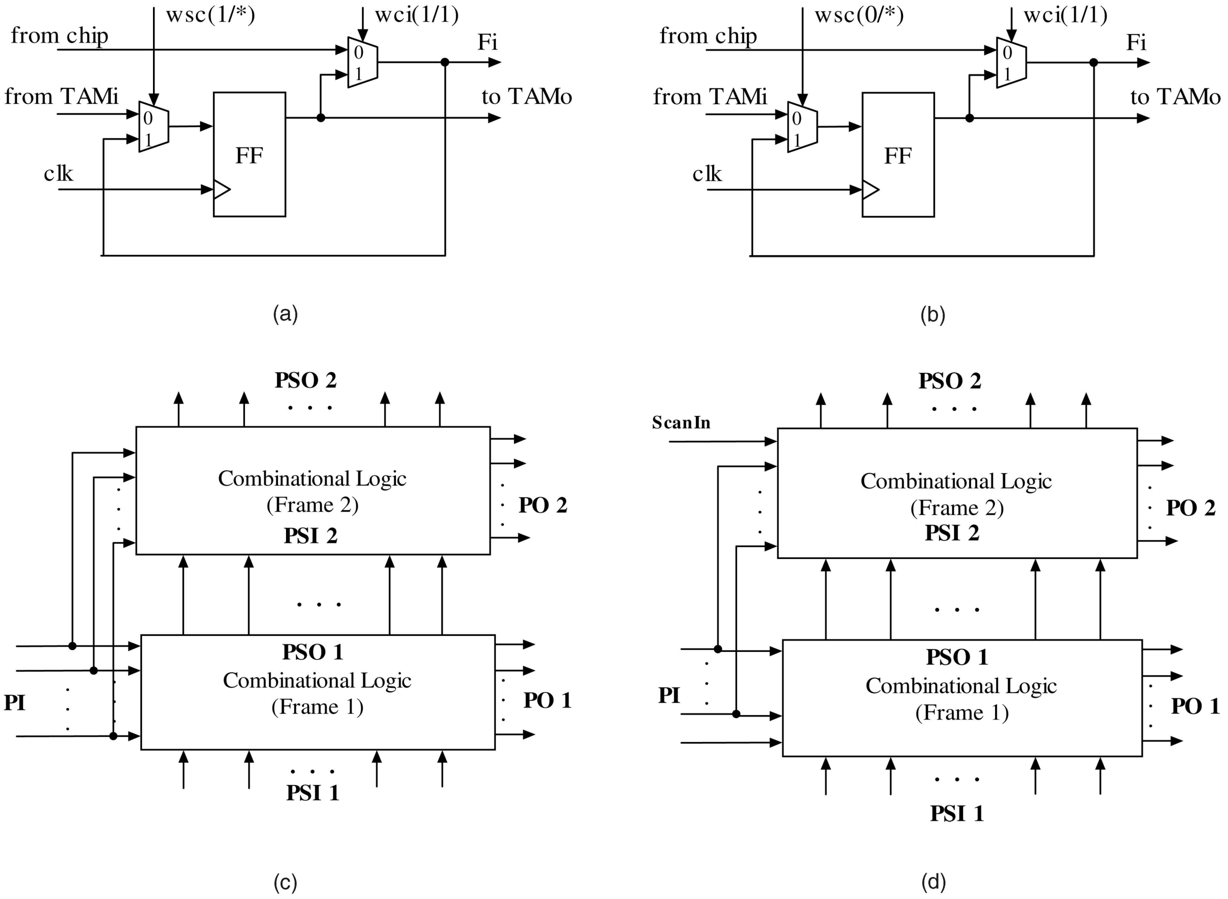
Fig. 3. Modular SOC test architecture categorization. (a) Fixed-width TAM bus. (b) Flexible-width TAM bus.

2.3 SOC Test Architectures

Based on the TAM lines assignment strategy, the modular test architectures (i.e., Test Bus and TestRail architectures) can be further categorized into the following two types [20]:

- *Fixed-width test bus architecture* (Fig. 3a), in which the total TAM width is partitioned among several test buses with fixed-width TAM wires. It operates at the granularity of TAM buses and each core in the SOC is assigned to exactly one of them.
- *Flexible-width test bus architecture* (Fig. 3b), in which TAM wires are allowed to fork and merge instead of just partitioning into TAM buses. It operates at the granularity of TAM wires and each core in the SOC can get assigned any TAM width as needed.

A vast body of research has been undertaken for optimizing both types of architectures [41]. For fixed-width test architectures, Iyengar et al. [17] first formulated the integrated wrapper/TAM cooptimization problem and broke it down into a progression of four incremental problems in order of increasing complexity. An integer linear programming (ILP) model was then presented to solve the problem. To decrease the CPU running time, the same authors combined efficient heuristics and ILP methods in [18]. Koranne [23] formulated the test scheduling problem as a network transportation problem and presented a 2-approximation algorithm to solve it. While the above approaches concentrate on Test Bus architecture, Goel and Marinissen [9] presented an efficient heuristic, $TR-Architect$, which works for both Test Bus and TestRail architectures. In [10], [11], $TR-Architect$ was extended to account for the wire length cost and test control, respectively. For flexible-width test architectures, Huang et al. [13] first mapped the test architecture optimization to the well-known two-dimensional bin packing problem and proposed a heuristic method based on the Best Fit Decreasing



(a)

(b)

(c)

(d)

Fig. 4. Wrapper input cells and combinational ATPG models for broadside testing when adapting existing approaches to control embedded core's primary inputs. (a) $NC-PI$ standard WIC control. (b) $SC-PI$ standard WIC control. (c) $NC-PI$ two-pattern ATPG model. (d) $SC-PI$ two-pattern ATPG model.

algorithm to solve it. Iyengar et al. [20] presented an improved heuristic for the rectangle packing problem, when cores are supplied with fixed-length scan chains. Next, in [19], the same authors extended their algorithm to incorporate precedence and power constraints, while allowing a group of tests to be preemptable. Later, in [21], they considered minimizing the tester buffer reloads and multisite testing. In [44], Zou et al. used sequence pairs to represent the placement of the rectangles, borrowed from the place-and-route literature and then employed a simulated annealing technique to find a better test schedule.

All the above modular test architectures and their optimization approaches consider only one-pattern tested cores. In the next section, we discuss how to adapt the existing methodologies for two-pattern tests, which motivates our proposed test architecture.

3 Motivation

No explicit mechanism for two-pattern test of core-based SOC is provided in the public domain. One possible method is to exploit SOC's architecture-specific information and to reuse on-chip functional interconnect as a test access mechanism [5], [8], [31], [42]. Regardless of their potential benefits in the long term, unless implemented automatically using a reliable test tool flow, these architecture-specific DFT methodologies do not provide reusability, flexibility, and interoperability. Therefore, we only consider how to apply two-pattern test for the modular bus-based SOC test architecture in this paper. The main challenge here lies in the fact that the cores' inputs are difficult to be fully controlled in consecutive clock cycles with regular IEEE Std. 1500 wrapper cells, thus resulting in fault coverage loss if broadside delay testing is used.

In broadside testing, the pseudo-input part of the excitation vector (i.e., the part that is loaded in the internal flip-flops) is generated through functional justification. However, in order to emulate the functional core behavior, we analyze two options for controlling the *PIs of the embedded core* that can reuse the existing algorithms for TAM design and test scheduling:

- *Non-Controlled Primary Inputs (NC-PI):* This test scenario assumes that PIs are scanned for fully controlling the initialization vector, however, they keep the same value (frozen) for the excitation vector by asserting wsc to 1 during the launch cycle; the control of the input WBR cells and the associated broadside ATPG model are shown in Fig. 4a and Fig. 4c, where labels on the wsc and wci multiplexer controls show the values during the launch/capture cycle;

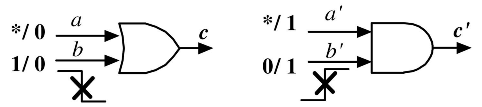


Fig. 5. Fault coverage loss with the $NC - PI$ and $SC - PI$ ATPG models.

- *Serially Controlled Primary Inputs (SC-PI):* After the PIs are scanned in for the initialization vector, in order to obtain the excitation vector, they are updated through an extra shift by resetting wsc to a 0 (using TAM data as input for the first WBR cell) during the launch cycle; the control of the input WBR cells and the associated broadside ATPG model are shown in Fig. 4b and Fig. 4d.

Since the $NC - PI$ and $SC - PI$ control mechanisms can reuse the existing TAM design and test scheduling algorithms, an obvious question is why do we need a new test architecture? The answer lies in the *quality of delay tests*. While the delay fault coverage loss caused by the connection between pseudo-output (PSO) 1 and pseudo-input (PSI) 2 *must* be ignored (these are redundant faults and will never be activated during functional operation, which is an advantage of broadside testing over skewed-load testing and enhanced-scan [33]), the coverage loss due to PI sharing between the two time frames (see both $NC - PI$ and $SC - PI$ in Fig. 4c and Fig. 4d, respectively) is unacceptable. In addition, another difficulty to using the $SC - PI$ control mechanism (but not the $NC - PI$ control mechanism) is that core providers might not be able to provide the $SC - PI$ test set because they need to tune their ATPG engine to support this scenario.

The limitation of the $NC - PI$ and $SC - PI$ control mechanisms can be easily illustrated by testing the b and b' inputs of the OR and AND gates for falling and rising transitions, respectively (see Fig. 5). For example, if OR input b precedes input a in the wrapper scan chain, then the falling transition on input b is untestable by $NC - PI$ and $SC - PI$ since either freezing ($NC - PI$) or shifting ($SC - PI$) the PI value, $b = 1$ required for initialization will conflict with $a = b = 0$ necessary for excitation in the following second capture cycle. One might argue that this problem can be solved by structural modifications (i.e., wrapper cell reordering), however, this may be prohibited due to routing constraints and it also brings additional design effort due to its test set-dependence. In addition, two-pattern test set compaction for $NC - PI$ and $SC - PI$ control mechanisms will introduce additional constraints (caused by freezing and shifting), which are difficult to satisfy when the care-bit density in each vector is increasing (i.e., when test set size decreases).

Based on the analysis in [22], if the system integrator wants to achieve 90 percent delay fault coverage, when there are five cores requiring delay tests, each core requires a delay fault coverage of about 98 percent. Therefore, to achieve such high fault coverage, a *Parallelly-Controlled Primary Inputs (PC-PI)* ATPG model, as shown in Fig. 6a, is necessary. This $PC - PI$ ATPG model guarantees that *any* primary input value can be justified for both initialization and excitation vectors. The easiest way to implement such an ATPG model is to double-buffer the core's wrapper input cells (called *enhanced wrapper input cell*), as depicted in Fig. 6b. When the CUT is in one-pattern test mode (e.g., stuck-at test), the load through two flip-flops will incur a needless clock cycle for each core input. Hence, a multiplexer, *TP mux*, is introduced to bypass $FF2$ in one-pattern test mode to reduce its loading time. As observed in Fig. 6b, each enhanced WIC introduces an extra flip-flop and an extra multiplexer (that is, an extra scanned flip-flop (SFF)) compared with regular WIC design. Since embedded cores have no pin constraints and may have a large number of I/Os, the enhanced WIC cell may incur a large DFT area overhead.

To lower the DFT area overhead without loss of fault coverage, in this paper, we describe an approach that is able to implement the PC-PI ATPG model with standard WIC design. The proposed PI control mechanism (as shown in Fig. 6c), in which the second test vector V_2 is justified through a parallel load from the core's *producer*[1] *cores*, leads to a $producer/CUT$ test architecture (discussed in detail in the next section). In each test session, the active cores are divided into $producers$ (cores that provide PI values for excitation) and $CUTs$ (cores tested in the current session through broadside test).

4 Proposed Architecture for Two-Pattern Test

The proposed approach considers, as a starting point, that each core in the SOC is part of the IEEE Std. 1500 architecture and, hence, it is 1500-wrapped [14]. In addition, user defined logic (UDL) is also treated as a 1500-wrapped core. In this section, we first introduce the generic two-pattern testing process using the proposed architecture and then we discuss the necessary DFT support for the proposed methodology.

4.1 The Two-Pattern Testing Process

The proposed approach uses the input WBR cells of the CUT to control the PIs of the launch vector and it exploits its producers' output WBR cells to control the PIs of the excitation vector. As shown in Fig. 7, this test session-level producer/CUT dichotomy leads to a division of the SOC's TAM into producer TAMs and CUT TAMs, which will be detailed in Section 5.

The proposed broadside two-pattern testing process can be explained as follows: The IEEE Std. 1500 instruction set is extended to support two custom modes, *LOADPROD* for producer cores and *TPTEST* for the two-pattern tested CUT. In the TPTEST mode, the loading of the initialization vector into the internal scan chains and the application of the initialization vector are done in the same way as for the INTEST mode. However, since TPTEST requires another capture cycle, an internal control mechanism for applying the excitation vector must be provided. On the one hand, the PSI of the excitation pattern (the internal scan chain part) is generated through functional justification. On the other hand, the PI part is provided using the functional

[1] For a given $Core_i$, the producers are the cores which feed its primary inputs in the normal (functional) mode.

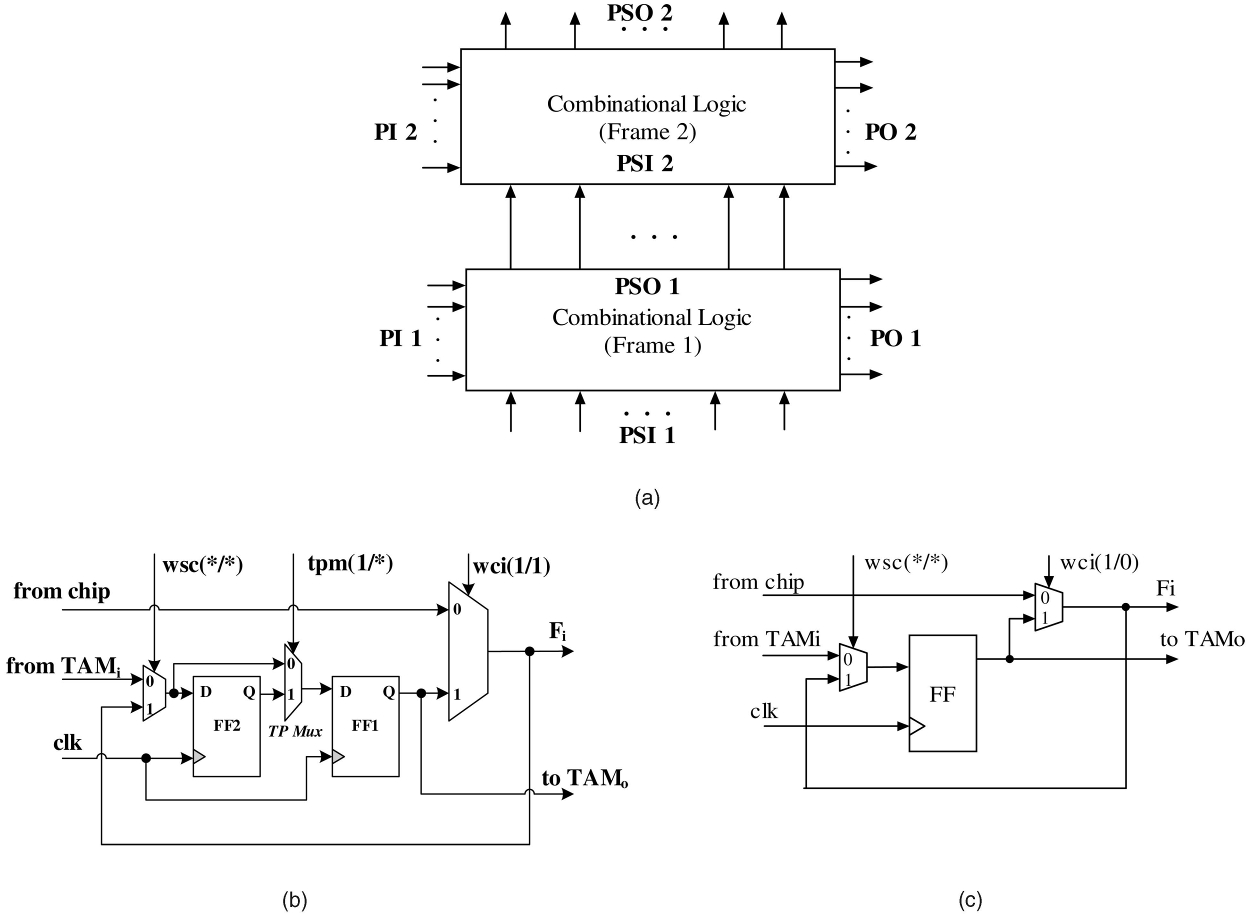


Fig. 6. PC-PI ATPG model and the two corresponding wrapper input cell designs. (a) $PC-PI$ two-pattern ATPG model [12]. (b) $PC-PI$ enhanced WIC control. (c) $PC-PI$ standard WIC control.

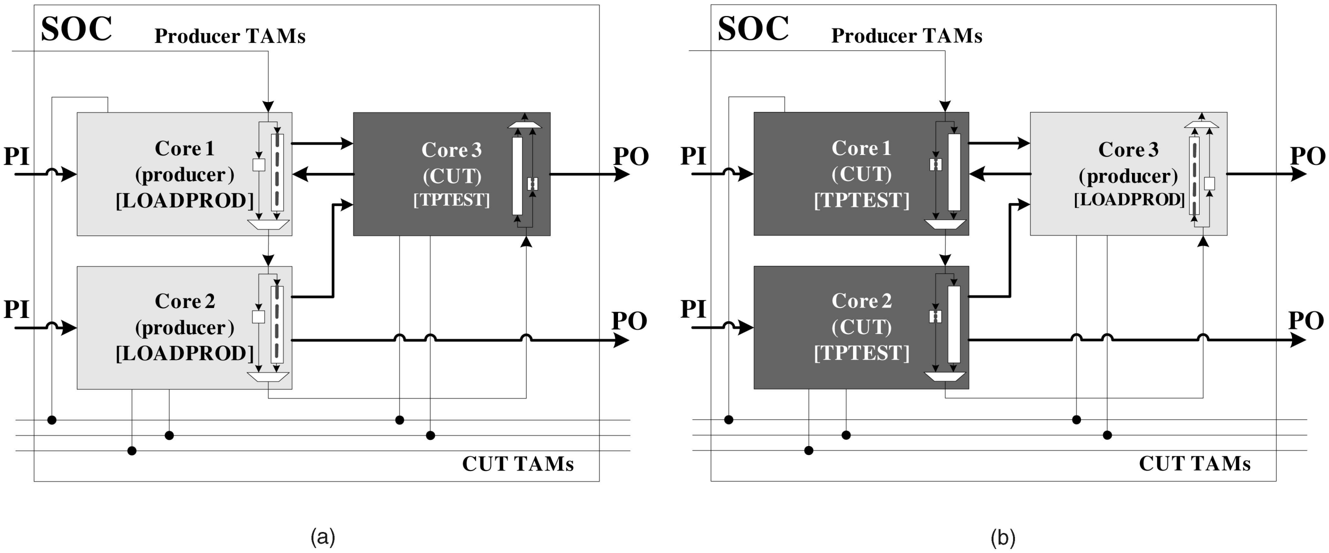


Fig. 7. Proposed approach for two-pattern test of core-based SOCs. (a) Session 1 for TPTEST of core 3. (b) Session 2 for TPTEST of cores 1 and 2.

inputs by setting the provider cores in the producer LOADPROD mode. Although simply configuring the producers in EXTEST mode can finish the same mission, it requires extra time to go through their input WBR cells. As a result, we propose that only the output WBR cells are connected as scan chains in the producer TAM lines to speed up this loading time. This can be seen in Fig. 7a and Fig. 7b, where two test sessions are required to test a hypothetical SOC for delay faults or stuck-open faults. To ensure two consecutive controllable PI vectors, the CUT's WBR multiplexer control signals are switching between the input WBR cells of the CUT for initialization and the output

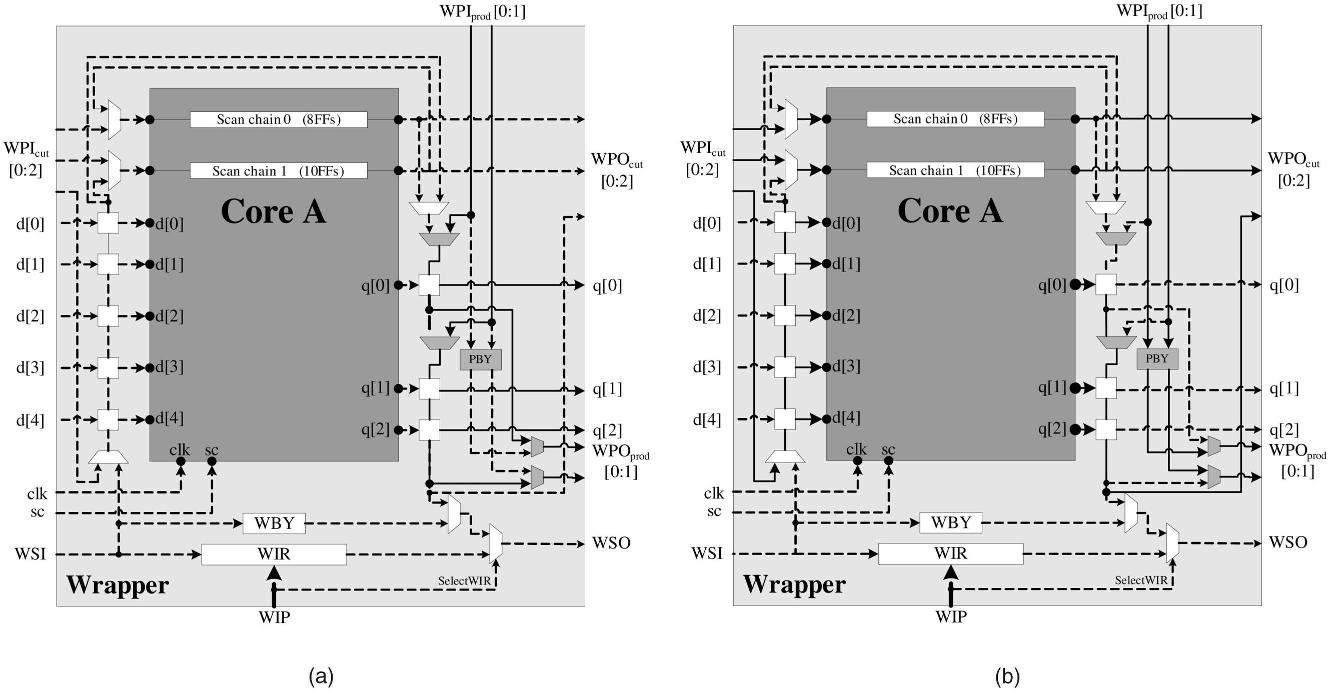


Fig. 8. An example producer core in (a) LOADPROD and (b) Bypass modes.

WBR cells of the producers for excitation. By exploiting the existing producers' output WBR cells for storing the PI part of the excitation vector, an *emulation of the enhanced-scan is provided only for the PIs of each core*. It is important to note that, to provide at-speed delay fault testing, the multiplexer control signals for WBR cells must switch at-speed, while, for stuck-open faults, they can switch at lower speed.

4.2 1500-Compatible Core Wrapper Design

Additional DFT hardware is necessary to decode the newly introduced instructions for two-pattern test and ensure the proper activities of the core wrapper in TPTEST or LOADPROD mode.

To support the proposed TPTEST mode, which applies two consecutive test patterns as in the PC-PI ATPG model, in the CUT core wrapper, we need to add some logic to decode the newly introduced TPTEST instruction. It is important to note, however, the at-speed switch of the *wci* signal (see Fig. 6c) for delay fault testing requires the core test wrapper to be controlled by a rated-speed clock signal. To support the proposed LOADPROD mode for the producer cores, however, in addition to the extra logic to decode the new LOADPROD instruction, the wrapper needs to be revised to be able to load test data into the output wrapper boundary cells. As shown in Fig. 8, for the example core A, two producer TAM lines are used to load test data into its output WBRs in LOADPROD mode (the real lines indicate which paths are enabled). In any other mode (Fig. 8b shows Bypass mode), the producer TAM lines bypass core A through PBY registers. Hence, four extra multiplexers and a 2-bits bypass register *PBY* are added to the wrapper in this example, which is much smaller when compared to the enhanced WIC design that implements an extra SFF for each core input terminal. In addition, buffers may need to be inserted to increase the driving strength of the *wci* signal for at-speed switching purposes. The DFT cost of the buffer insertion is quite small when compared to the enhanced WIC design. It should also be noted that when, in between cores, there is some unwrapped logic in addition to wires (this situation is not the focus of this paper), we can let the CUT inputs that are not driven from producers directly be wrapped with enhanced wrapper input cells and let the other inputs be wrapped with IEEE Std. 1500 regular wrapper input cells. The above does not affect the operation of the proposed test architecture.

In summary, when compared to a standard IEEE Std. 1500 wrapper implementation (i.e., INTEST, EXTEST, and BYPASS), only several logic gates to decode the new TPTEST and LOADPROD instructions, a few multiplexers to chain the output WBR cells in LOADPROD mode, and a small number of buffers for *wci* signal at-speed switching need to be introduced to support two-pattern test, which is usually negligible.

5 TAM Design and Test Scheduling

Having introduced the $producer/CUT$ test architecture supported by the new LOADPROD and TPTEST instructions, the $PC - PI$ ATPG model, and the wrapper design required on the core provider's side, this section concentrates on the TAM design and test scheduling for two-pattern test, which is necessary on the system integrator's side.

5.1 Test Conflicts

It is known that, in the INTEST mode, embedded cores can be tested concurrently as long as they use different TAM lines; however, this is not the case for the proposed TPTEST mode because the CUT needs the cooperation of its producers to supply the excitation vector. Two new types of test conflicts are introduced as follows:

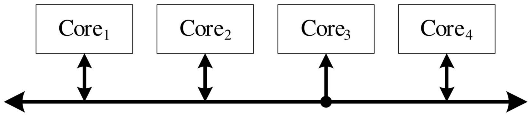


Fig. 9. Test conflicts for multiple cores on the same bus.

- *Producer-CUT Conflict*: Producers and the CUT **cannot** be tested at the same time. This is because, the producer needs to utilize its output WBRs to capture its test responses; however, at the same time, the CUT needs the producer's output WBRs to provide test stimuli. If they are tested concurrently, the test data will be corrupted. For example, in Fig. 7, Core 3 should not be tested with Core 1 and Core 2 concurrently.
- *Shared-Producer Conflict*: Two cores which directly connect to the same producer(s) for the excitation vector **cannot** be tested at the same time; this is because both of them require the output WBRs of the same producer(s) to provide the test stimuli. For example, in Fig. 7, if Core 2 also gets inputs from Core 3, then Core 1 and Core 2 should not be tested concurrently.

When two cores are not directly connected, but they communicate through functional buses, they may imply the test conflicts described above and, hence, **may not** be able to be tested concurrently. For example, as shown in Fig. 9, since Core 1, Core 2, and Core 4 all can transmit data to the bus, it is possible to select any of them to be the producer of the other cores on bus to provide the excitation vector. We name this type of producer a *bus producer*. A different selection of bus producers will generate different test conflicts between each other, as shown in the following example.

Example 1. Suppose $Core_2$ is selected to be the producer of $Core_1$ and $Core_4$ serves as the producer of $Core_3$, obviously $Core_1$ and $Core_2$, $Core_3$ and $Core_4$ cannot be tested concurrently because of the producer-CUT test conflict. However, $Core_1$ and $Core_3$ can be tested at the same time; this is because their producer ($Core_2$ and $Core_4$, respectively) can be loaded at the same time and *only* during the launch/capture cycles does the functional bus need to be shared. Therefore, the additional testing time penalty is determined by serializing the capture cycles for $Core_1$ and $Core_3$ tested through the functional bus. The added couple of clock cycles cause negligible penalty on scan time. However, if both $Core_1$ and $Core_3$ get test stimuli from $Core_2$, there is shared-producer conflict between them and they cannot be tested concurrently.

If there are test conflicts between cores, then these cores are called *incompatible cores* and they cannot be scheduled concurrently during test. From Example 1, different bus producer configurations will generate different test conflicts between them and, hence, influence the SOC test schedule. We propose the *Assign_Bus_Producer* algorithm for the selection of bus producers, as shown in Fig. 10. The main idea is to decrease the total test conflicts of the SOC. The first while loop (lines 2-8) finds those cores that are already incompatible on the bus and assigns bus producers to each other. This will not introduce any more test conflicts in the SOC. Afterward, for the remaining cores that do not have an already incompatible producer on the same bus, the second while loop (lines 9-16) simply assigns bus producers for them in sequence. After analyzing the test conflicts between each core, we construct a test incompatibility graph (TIG) by treating each core as a node and by adding an edge between two nodes if the cores are incompatible. This TIG is used in the test scheduling Algorithm (*Adapted_TAM_Schedule_Optimizer*) described in the following sections.

```
Algorithm 1 - Assign_Bus_Producer

INPUT: $C$ //Core list on bus
OUTPUT: $C_{prod}$ //Bus producer for each core on bus

1. $C' = C$;
2. While ($true$) {
3.    if ($\exists$ $Core_i \in C'$ has incompatible $Core_j \in C$) {
4.       Set $Core_i, Core_j$ as the bus producer for each other;
5.       if ($Core_j \in C'$) {
6.          $C' = C' - \{Core_i, Core_j\}$;
7.       } else { $C' = C' - \{Core_i\}$; }
8.    } else { break; }
.  }
9. While ($C' \neq \emptyset$) {
10.   if ($Core_i$ is the last core in $C'$) {
11.      Pick any $Core_j \in C$ as its producer;
12.      $C' = \emptyset$;
13.   } else {
14.      Pick next core $j \in C'$;
15.      Set $Core_i, Core_j$ as the bus producer for each other;
16.      $C' = C' - \{Core_i, Core_j\}$; }
.  }
17. return $C_{prod}$;
```

Fig. 10. Procedure for assigning producer cores on functional buses.

5.2 TAM Division into Producer/CUT Groups

Producers and CUTs should be fed from **two separate** TAM groups, otherwise additional *indirect* test conflicts may be introduced. This can be seen from the following example:

Example 2. Suppose the test data is transferred using shared TAM lines between producers and CUTs during two-pattern test. $Core_1$ and $Core_2$ are connected to separate TAM lines, however, $Core_1$ shares its TAM lines with $Core_3$, which is a producer to $Core_2$. To test $Core_2$ in TPTEST mode, we need $Core_3$ in LOADPROD mode since it shares TAM lines with $Core1$. Loading a pattern in $Core_1$ is prohibited at this time, although it uses separate TAM lines to $Core_2$. Hence, this *indirect* resource conflict leads to test conflict between $Core_1$ and $Core_2$.

A neat solution to this problem, which is important in particular for complex SOCs with a large number of two-pattern tested cores, is to divide the TAM lines into two groups: G_{prod} for loading the PIs of the excitation patterns in the producers' outputs and G_{CUT} for loading the initialization patterns and unloading the test responses from the CUT. In this way, for the above example, $Core_3$ in LOADPROD mode does not affect loading $Core_1$ since they use TAM lines from different TAM groups (hence, no additional test conflict between $Core_1$ and $Core_2$ exists).

For the G_{CUT} group, a flexible-width test bus architecture is used to support efficient test scheduling. For the G_{prod} group, however, we use a daisychain architecture, i.e., long scan chains are constructed over the output terminals of the cores that serve as producers during test, as shown in Fig. 7. Bypasses are introduced in order to shorten the loading time because only a few cores serve as producers at a specific test session. The main reason for using the daisychain architecture for the G_{prod} group is to simplify the control complexity. When a producer core is in the LOADPROD mode, the producer TAM lines go through the core's output wrapper boundary cells; otherwise, they go through bypass registers (note that it is unnecessary to introduce extra producer bypass instruction because it is compatible with the standard 1500 Bypass mode). As a result, although the two-pattern test of CUT involves several producers, these producers can be controlled by the LOADPROD instructions independently and no extra control signals need to be supplied. In addition, the daisychain architecture for G_{prod} can almost always give a near optimal loading time for a given producer TAM width W_{prod}. Suppose the number of the outputs of a producer is N_o, then its loading time will be $\lceil \frac{N_o}{W_{prod}} \rceil$. As long as $W_{prod} \leq N_o$ (which is realistic in most of the cases), there is no waste for G_{prod} TAM resources except the few bypass cycles. This leads to a near optimal loading time for producers in each test session.

5.3 Two-Pattern Test Scheduling

The introduction of producer/CUT model and TAM division into the G_{prod} and G_{CUT} groups, leads to new test scheduling algorithms, as explained in this section. Note that this paper does not directly address the design of hierarchical TAMs since we assume that the SOC hierarchy is flattened during TAM design (hierarchical cores are considered as being at the same level in test mode). This paper also does not consider test scheduling constraints introduced by precedence relationship, preemption, and power, as in [16].

Problem P_{TP-opt}. *Given the test set parameters for each core (including the number of primary inputs, primary outputs, bidirectional I/Os, test patterns and scan chains, and each scan chain length), the functional relationships between cores R, the total TAM width* W_{ttl} *for the SOC, determine the width of each TAM group (*W_{prod}*,* W_{CUT} *corresponding to* G_{prod}*,* G_{CUT}*), the assigned TAM width and the wrapper design for each core and a test schedule for the entire SOC such that: 1) the total number of TAM lines used at any time does not exceed* W_{ttl} *and 2) the overall SOC testing time is minimized.*

```
Algorithm 2 - TAM_Division_And_Schedule
INPUT: $C_{set}$, $R$, $W_{ttl}$
OUTPUT: $W_{prod}$, $W_{CUT}$, $schedule$, $TAT_{soc}$
1. $Assign\_Bus\_Producer(C_{set})$;
2. $TIG = Construct\_TIG(R)$;
3. For $W_{CUT}$ from $W_{ttl} - 1$ downto 1 {
4.     $W_{prod} = W_{ttl} - W_{CUT}$;
5.     $testing\_time = Adapted\_TAM\_Schedule\_Optimizer($
.                  $C_{set}, TIG, W_{prod}, W_{CUT})$;
6.     $minTime$ = min{all $testing\_time$};
7.     Record $W_{prod\_min}$ and the associated $schedule$;
.  }
8. $W_{prod} = W_{prod\_min}$;
.  $W_{CUT} = W_{ttl} - W_{prod}$;
.  $TAT_{soc} = minTime$;
9. return $W_{prod}$, $W_{CUT}$, $schedule$, $TAT_{soc}$;
```

Fig. 11. Pseudocode for wrapper/TAM cooptimization.

The proposed algorithm *TAM_Division_And_Schedule* to solve P_{TP-opt} is shown in Fig. 11. The inputs are the set of cores (C_{set}), the total TAM width (W_{ttl}), and the functional interconnect relationship between cores (R). The outputs are the number of TAM lines allocated to each group W_{prod} and W_{CUT}, the wrapper design for each core, SOC test schedule $schedule$, and the overall test application time TAT_{soc}. The optimal TAM division, i.e., the combination of W_{prod} and W_{CUT} that gives the minimum TAT_{soc} of the SOC, is acquired through enumeration. The enumerative algorithm begins by assigning bus producers for those cores on buses (line 1), then, based on the test conflicts determined by the functional interconnect relationship between cores (R), a test incompatibility graph (TIG) (discussed in Section 5.1) is created (line 2). Next, inside the loop (lines 3 to 7), the algorithm will find the optimal TAM division and the system TAT TAT_{soc} by enumerating W_{CUT} from the maximum possible value $W_{ttl} - 1$ down to 1.

It should be noted that, during the enumeration process, we do not need to do TAM design for the G_{prod} group because it is already fixed using the daisychain architecture. To optimize the G_{CUT} TAM group, we adapt an existing generalized rectangle packing algorithm *TAM_Schedule_Optimizer* [20]. TAM_Schedule_Optimizer first finds out the pareto-optimal TAM widths for all cores. Next, a "preferred TAM width" [20] for each core is identified from these pareto-optimal TAM widths such that the core's TAT is within a small percentage of its testing time at a maximum allowable TAM width. The test for each core is then scheduled using the preferred width as long as there are enough TAM lines available. If the number of available TAM lines is insufficient to schedule any new tests, the resulting idle time is filled using several heuristics that insert tests to minimize the idle time. Whenever a currently running test completes, the number of available TAM lines is incremented and the algorithm repeats the scheduling process for the remaining tests. This is a rather simple description of the algorithm. The reader is referred to [20] for more details and terminology. The key novel feature in

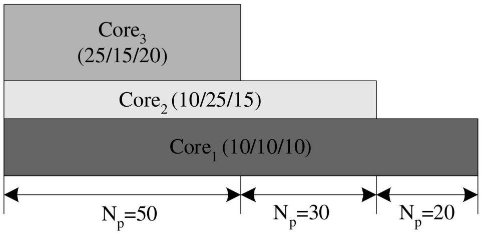


Fig. 12. Loading time for different patterns ($L_{prod}/si/so$).

our approach is that, due to the usage of daisychain architecture for loading producers' outputs, a *dynamic adaptation* of the existing algorithms is necessary, as discussed in the following paragraphs.

Dynamic Rectangle Representation: For one-pattern test of a 1500-wrapped core, if the assigned TAM width is given, the TAT to apply the entire test set T_p is determined by (1) [27], in which $s_i(s_o)$ is the longest wrapper scan-in (scan-out) chain for the core and p is the number of test patterns. When using the $Design\_wrapper$ algorithm from [17] for wrapper optimization, $s_i(s_o)$ has a fixed value for a given TAM width and, hence, the core test can be represented as a *static* rectangle.

$$T_p = (1 + max\{si, so\}) \times p + min\{si, so\}. \tag{1}$$

However, when reusing functional interconnect to transfer test data for two-pattern tested cores, its TAT not only depends on the time to load its own producers (L_{prod}), wrapper scan-in chains (si), and unload its wrapper scan-out chains (so). The TAT also depends on the time necessary to load/unload all the other concurrently tested cores' producers and wrapper scan in/out chains. To keep the control and computational complexity low, we propose *aligning* test patterns for all of the concurrently tested two-pattern tested cores. That is, if several two-pattern tested cores are scheduled at the same time, then the overlapped test patterns for these cores will have the same start times and have the same loading/unloading time for each pattern, which we call the $LoadSize$.

$$LoadSize = max\Big\{\sum L_{prod} + L_{bypass} - 1, max\{si\}, max\{so\}\Big\}, \tag{2}$$

in which L_{bypass} is the number of bypass clock cycles on the unused producers. Obviously, the core which needs the least time to load its test stimuli will wait for all the currently scheduled cores to complete loading, thus leading to an increase in the TAT of other cores currently under test. Hence, if, for the given core, the test schedule changes s times, then, for each subset of patterns p_s (for the s distinct divisions of the time allocated to the given core), the testing time will be computed based on the two-pattern tested cores scheduled in each of these s divisions. Since the loading time for each tested core is variable with its schedule, the rectangles cannot be precomputed in the proposed TPTEST mode. This can be observed from the following example.

Example 3. Suppose $Core_1$, $Core_2$, and $Core_3$ are two-pattern tested cores and they are scheduled as in Fig. 12, the loading time for their producers and internal scan chains is shown in the figure. Since the producers and CUTs are connected using the daisychain architecture, without considering L_{bypass}, for the first 50 patterns, the $LoadSize$ for all the three cores will be

$$max\{L_{prod\_1} + L_{prod\_2} + L_{prod\_3} - 1, si\_1, si\_2, si\_3, so\_1, so\_2, so\_3\} = 44$$

clock cycles. However, for the next 30 patterns, once the test for $Core_3$ has been completed, the $LoadSize$ will be $max\{L_{prod\_1} + L_{prod\_2} - 1, si\_1, si\_2, so\_1, so\_2\} = 25$ clock cycles. The same reasoning is applied for the last 20 patterns, when $Core_1$ is not concurrently tested with any other cores, where the $LoadSize$ will be 10 instead.

```
Algorithm 3 - Adapted_TAM_Schedule_Optimizer

INPUT: $C_{set}, TIG, W_{prod}, W_{CUT}$
OUTPUT: $schedule, TAT_{soc}$

1. Compute collection $R_o$ of rectangles for one-pattern tested core set $C_o$;
2. Initialize($C_o$, $d$, $p$);
3. Set $C_{unfinished} = C_{set}$; $w\_avail = W_{CUT}$; (see [20])
4. While $C_{unfinished} \neq \emptyset$ {
5.   if $w\_avail > 0$ {
6.      Find unscheduled two-pattern tested core set $C'_t$;
7.      Compute collection $R'_t$ of dynamic rectangles for $C'_t$;
8.      Initialize($C'_t$, $d$, $p$);
9.      Schedule compatible one-pattern tested cores that can be assigned
.       preferred TAM width or two-pattern tested cores; (see [20])
10.     Schedule compatible one-pattern tested cores that can use the
.       resulting idle TAM wires; (see [20])
11.     Update $L_{prod}$, $loadSize$;
12.     Update $test\_time$ for scheduling two-pattern tested cores;
13.  } else {
14.     Update $next\_time$;
15.     Finish the scheduling core test $C_i$ with ending time $next\_time$;
16.     Update $this\_time$;
17.     $w\_avail += w_{tam\_C_i}$;
18.     $C_{unfinished} -= \{C_i\}$;
.    }
. }
19. return $schedule$, $TAT_{soc}$;
```

Fig. 13. Procedure for test scheduling with given widths for each TAM group.

Adapted Dynamic Rectangle Packing: Fig. 13 shows the pseudocode for algorithm *Adapted_TAM_Schedule_Optimizer*. The algorithm takes the core list C_{set}, TIG, and the TAM division as inputs and it generates the $schedule$ for each core and the SOC testing time TAT_{soc}. The proposed algorithm is based on a generalized rectangle packing algorithm [20] and we only show the differences with respect to the original algorithm.

As described earlier, for two-pattern tested cores, the test **cannot** be precomputed and represented as a static rectangle. Its TAT (the width of the rectangle) varies with its schedule and, hence, its rectangle representation is computed dynamically (line 7). For the same reason, there

TABLE 1
Comparison of the Three Two-Pattern ATPG Fault Models on ISCAS89 Benchmark Circuits

			Fault Coverage (%)			Test Pattern Count		
circuit	N_{in}	N_{ff}	$NC-PI$	$SC-PI$	$PC-PI$	$NC-PI$	$SC-PI$	$PC-PI$
s1243	17	74	73.82	83.91	88.86	181	187	211
s1494	7	6	77.33	91.17	94.26	155	172	188
s5378	35	179	74.54	87.43	88.05	374	390	437
s9234	36	211	79.39	84.97	90.38	400	351	424
s13207	62	638	75.64	82.84	83.09	676	777	796
s15850	77	534	65.42	79.65	83.96	610	756	862
s35932	35	1728	80.58	85.21	86.54	130	124	131
s38417	28	1636	89.44	90.33	90.39	1698	1716	1753
s38584	38	1426	75.05	91.21	92.09	2137	1977	2000

are also no "preferred TAM widths" for two-pattern tested cores. In [20], the procedure *Initialize* is used to compute the preferred width for each core. The parameters d and p are used to select appropriate "preferred width" for each core and are usually manually selected for SOCs with different available TAM widths to get a better result. Since we need to call this procedure many times with different W_{CUT} (see Algorithm 1), it is unlikely that a manual selection will lead to an optimal value. Consequently, in our implementation, we have fixed the two parameters to $d = 2$ and $p = 1.0$. This may result in a different schedule and a slightly longer testing time in some cases when compared to the result from [20]. Whenever a core is scheduled (lines 9 and 10), the available TAM width w_avail will be deducted with the value of the assigned G_{CUT} TAM width for the core. Once a two-pattern tested core is scheduled, L_{prod}, $loadSize$ for the currently scheduled cores in TPTEST mode needs to be updated (line 11) and their TATs are recalculated (line 12). Once there is no available TAM resources for current test session (i.e., $w\_avail = 0$), the currently scheduled core with the minimum TAT will complete (line 15), its TAM resources will be released, and the algorithm will try to find another unscheduled core which can use the freed TAM lines.

6 Experimental Results

The proposed two-pattern test architecture is based on the observation that the $PC-PI$ ATPG model offers better test quality than the $NC-PI$ and $SC-PI$ ATPG models. To prove this, we first compare the fault coverage of the three two-pattern ATPG models by performing experiments on the ISCAS89 benchmark circuits [3]. Then, to investigate the implication of the proposed approach on the DFT area overhead and the SOC TAT, experiments are also carried out on a revised version of ITC'02 SOC benchmark circuits [28] whose specifications detailed in Section 6.1. We compare the experimental results with its counterpart when enhanced WICs are used for two-pattern test. As described in Section 5, the total TAM lines are divided into producer/CUT TAM groups in the proposed test architecture. Since different divisions (configurations) will generate different test schedules, we need to obtain the optimal configuration which leads to the minimum TAT. The above issues are investigated in the following four experiments:

Experiment 1 compares the fault coverage of the NC-PI, SC-PI, and PC-PI two-pattern ATPG models (Section 6.2).

Experiment 2 discusses the DFT area savings of the proposed test architecture (Section 6.3).

Experiment 3 illustrates the variable TAT with different producer/CUT TAM configurations (Section 6.4).

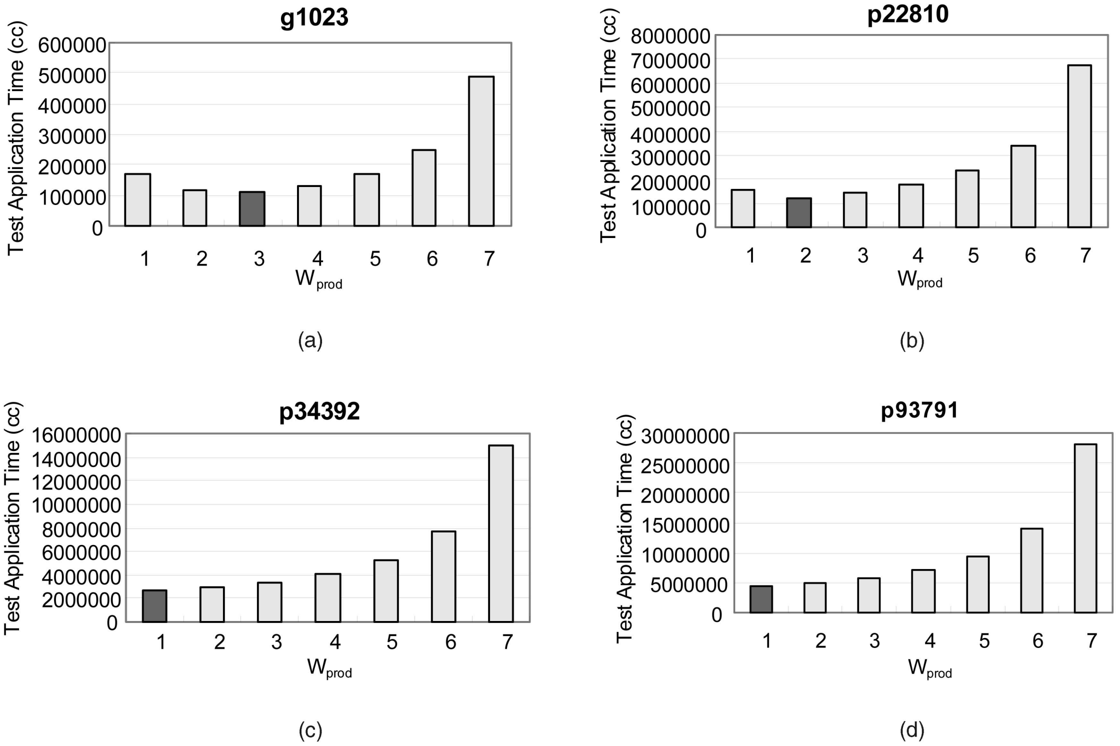
Experiment 4 describes the TAT changes when using the new TPTEST methodology (Section 6.5).

6.1 SOC Specifications

Four SOCs, g1023, p22810, p34392, and p93791, which are originally part of the ITC'02 *SOC test benchmarking initiative* [28], are used in experiments 2, 3, and 4. g1023 is a comparatively small hypothetical SOC, while p22810, p34392, and p93791 are large industrial SOCs. Since the functional interconnects are not provided in the benchmark files, we have decided to randomly generate them to support the proposed approach [40], including the direct connection between cores and functional buses. We have assumed that the SOCs have $Round(\frac{N_c}{10})$ buses and each bus has a random number $p\ (3 \leq p \leq min(N_c, 8))$ of cores attached to it, where N_c is the total number of cores in the SOC. In addition, all cores on buses are assumed to be able to transfer data to and from the bus and, hence, each one of them can be a bus producer to others. We have also assumed that every core has a random number of $q(1 \leq q \leq 3)$ producers. In addition, we assume that all the cores with internal scan chains are tested using the proposed TPTEST modes, while the remaining nonscanned cores are tested using INTEST mode. Hence, 12 of 14 cores in g1023, 22 of 28 cores in p22810, 4 of 19 cores in p34392, and 13 of 32 cores in p93791 are selected to be tested in TPTEST mode. In addition, we assume that the number of test patterns is equal to the one provided in [28] when cores are two-pattern tested. It should be noted, however, that, in reality, the number of patterns for delay faults is usually much higher than when targeting single stuck-at faults.

6.2 Experiment 1: Fault Coverage Comparison

Table 1 presents the fault coverage and test pattern count for the three two-pattern ATPG models (acquired from a commercial ATPG tool—*Synopsys TetraMAX*). In addition, the number of inputs and internal flip-flops of the circuit are also shown in the table, denoted as N_{in} and N_{ff}, respectively. Since TetraMAX does not automatically support the $SC-PI$ ATPG model, we revised the circuit so



(a)

(b)

(c)

(d)

Fig. 14. SOC TATs with variable producer/CUT TAM width configurations. (a) TAT for SOC g1023. (b) TAT for SOC p22810. (c) TAT for SOC p34392. (d) TAT for SOC p93791.

that all the original primary data inputs (i.e., except clock and reset signals, if any) work as an internal shift register in the functional mode of the revised circuit and we introduce a single data input to the circuit to be hooked up at the beginning of the shift register. By revising the circuit as above, the initialization/excitation pair is acquired the same way as for the $SC - PI$ ATPG model.

It can be observed from the table that the average fault coverage improvement of the $PC - PI$ ATPG model over the $NC - PI$ and the $SC - PI$ ATPG models is $12 \sim 13$ percent and $2 \sim 3$ percent, respectively. As a result, to ensure a high test quality, the $PC - PI$ ATPG model is preferred. We can also observe that, for some circuits, the fault coverage difference between the $SC - PI$ and the $PC - PI$ ATPG models is quite small (e.g., for s38417). Therefore, for such circuits, the system integrator can select using the $SC - PI$ ATPG model in order to reduce the DFT area overhead; however, the ATPG tools utilized by the core provider to generate their test patterns must support this $SC - PI$ ATPG model. If this is the case, the system integrator can select some cores tested using the $PC - PI$ ATPG model, while others are tested using the $SC - PI$ ATPG model, based on a quick analysis of their fault coverage difference. It is important to note that any loss in coverage for $PC - PI$ will not influence the yield since the undetectable faults can never affect the native mode, which is not the case for $NC - PI$ and $SC - PI$, as stressed before in Section 3. Besides, if the system integrator receives a core with the $SC - PI$ test set from a core provider, then the order of the input WBR cells (including the number of TAM lines) will be predefined, which will impose additional constraints on the TAM design.

6.3 Experiment 2: DFT Area Savings

The area of the core wrapper is mainly determined by the size of the WBR cells. As discussed in Section 3, the enhanced WIC design introduces an extra SFF when compared to the standard WIC design. The area of a typical SFF implementation is about 10 equivalent 2-input NAND gates. Therefore, the DFT savings of the proposed architecture can be calculated as $\sum_i^{N_{tp}} \{N_{in}^i + N_{bi}^i\} \times 10$, in which N_{tp}, N_{in}^i, and N_{bi}^i are the number of two-pattern tested cores, the number of inputs for two-pattern tested core $Core_i$, and the number of bidirectionals for $Core_i$. Based on the above formula, using the proposed test architecture, SOC g1023, p22810, p34392, and p93791 save $14,910$, $23,820$, $4,840$, and $31,320$ equivalent 2-input NAND gates, respectively, when the introduced DFT logic of the proposed methodology is neglected (in the range of hundreds of gates). Because these savings do not come at no expense, the implications on the SOC TAT are discussed in the following sections.

6.4 Experiment 3: Optimal Producer/CUT TAM Configuration

Fig. 14 presents the TATs of the four SOCs, for different widths of the producer TAM (W_{prod}), when the total TAM width W_{ttl} is fixed to 8. To give an exact TAM width division, the functional interconnects are fixed in this experiment (i.e., we have randomly generated the functional interconnect only once). It can be seen that the TAT of g1023 is minimum when W_{prod} is 3, the TAT of p22810 is minimum when W_{prod} is 2, while the TAT for p93791, and p34392 is minimum when W_{prod} is 1. This variation is due to the relationship between the total number of the producers' outputs and the internal SFF in the SOCs. On the one hand,

TABLE 2
TAT Comparison of the Two Two-Pattern Test Methodologies for SOC g102

	g1023				
	Enhanced WIC	Regular WIC			
W_{ttl}	T(cc)	T_{ave}(cc)	$T_{\max}$(cc)	$T_{\min}$(cc)	ΔT(%)
8	76218	126976	152613	107117	+66.60
16	42106	65326	75813	57149	+55.15
24	27906	43906	54325	38518	+57.34
32	21967	33971	41268	28505	+54.65
40	17925	29057	34622	24838	+62.10
48	14794	27236	33549	22353	+84.10
56	14794	26728	33287	21676	+80.67
64	14794	26538	33223	21275	+79.38

TABLE 4
TAT Comparison of the Two Two-Pattern Test Methodologies for SOC p34392

	p34392				
	Enhanced WIC	Regular WIC			
W_{ttl}	T(cc)	T_{ave}(cc)	$T_{\max}$(cc)	$T_{\min}$(cc)	ΔT(%)
8	2198975	2715071	2752472	2708766	+23.47
16	1077812	1245226	1322366	1114607	+15.53
24	838643	1018869	1164818	855407	+21.49
32	544579	863964	1140994	704813	+58.65
40	544579	836397	1140994	567044	+53.59
48	544579	836274	1140994	567044	+53.56
56	544579	836274	1140994	567044	+53.56
64	544579	835413	1140994	567044	+53.41

in the case of g1023, the number of SFF is comparable to the number of the producers' outputs, hence, a large amount of TAT is necessary to load producers' outputs, thus leading to a higher number of producer TAM lines. On the other hand, for p93791 and p34392, the SFF number is significantly larger than the number of producers' outputs. As a result, the time necessary to load the internal scan chains dominates the SOC TAT and, hence, only one TAM line is necessary to load the producers' outputs. The number of SFFs in SOC p22810 is larger than the producer's outputs, but the difference is not as large as in the case of p34392 and p93791. Hence, two TAM lines are used to load producers' outputs to give the optimum SOC TAT. It should also be noted that the SOC TAT variation with W_{prod} is a convex function based on our observation, i.e., it decreases until it reaches its minimum for the *optimal* W_{prod}, after which point, if W_{prod} is further increased, then the TAT will grow as well. This is because, once sufficient TAM resources are used to load producers' outputs, the SOC TAT will be dominated by the loading of the internal scan chains (i.e., the G_{CUT} group). Hence, the further decrease of W_{CUT} will obviously lead to an increase of the overall TAT of the SOC.

6.5 Experiment 4: Test Schedule and Test Application Time

Table 2, Table 3, Table 4, and Table 5 present results for TPTEST of the four benchmark SOCs g1023, p22810, p34392, and p93791 when varying the total TAM width W_{ttl} (note that only results for the optimal TAM division are reported). Since the SOC TAT is affected by the functional interconnects, we ran the algorithm for 100 randomly generated interconnects. T_{ave}, $T_{\max}$, and $T_{\min}$ denote the average, maximum, and minimum TAT separately. The percentage change in TAT using the proposed test architecture is calculated using the formula $\Delta T(\%) = \frac{T_{ave}-T}{T} \times 100$, where T is the result for two-pattern test with enhanced WICs, using the algorithm from [20].

It can be seen that the average SOC TAT increases about 68 percent for SOC g1023, 43 percent for SOC p22810, 42 percent for SOC p34392, and 14 percent for SOC p93791, respectively. The increase is due to: 1) W_{prod} TAM lines used to load the excitation vector and 2) test resource conflicts between cores, as described in Section 5.1. Note that the increase varies based on the SOC structure (including the functional interconnects, the number of cores in the TPTEST mode, and the core sizes).

Having reported the average TAT increase for the four SOCs with 100 random functional interconnects, we present the different test schedules of the four SOCs for INTEST and TPTEST in Fig. 15. To get the exact schedule, the functional interconnect is also fixed in this experiment and only the schedule with optimal producer/CUT TAM width configuration is shown. For SOC g1023, the total TAM width is 16 and four TAM lines are used to load producers' outputs in the TPTEST mode. This obviously increases TAT, in addition to the reason that the loading time for each core is not solely dependent on its assigned TAM width, but also depends on the other producers of the other cores that are scheduled at the same time. For SOC p22810 with a total

TABLE 3
TAT Comparison of the Two Two-Pattern Test Methodologies for SOC p22810

	p22810				
	Enhanced WIC	Regular WIC			
W_{ttl}	T(cc)	T_{ave}(cc)	$T_{\max}$(cc)	$T_{\min}$(cc)	ΔT(%)
8	973995	1251036	1393431	1171321	+28.44
16	504440	653548	742651	607702	+29.56
24	368493	491847	559716	441247	+33.48
32	281438	397183	484224	332677	+41.13
40	241237	337832	410470	291417	+40.04
48	200595	301209	375937	250480	+50.16
56	174485	286088	360592	248916	+63.96
64	174485	278992	360592	239484	+59.89

TABLE 5
TAT Comparison of the Two Two-Pattern Test Methodologies for SOC p93791

	p93791				
	Enhanced WIC	Regular WIC			
W_{ttl}	T(cc)	T_{ave}(cc)	$T_{\max}$(cc)	$T_{\min}$(cc)	ΔT(%)
8	3684114	4228052	4323478	4198829	+14.76
16	1888950	2227268	2386166	2107380	+17.91
24	1302093	1506468	1679554	1287985	+15.70
32	1066517	1206364	1275858	1168159	+13.11
40	880381	1057245	1188497	898920	+20.09
48	694260	841771	979567	754301	+21.25
56	627575	641982	669208	624541	+2.30
64	580610	623852	643842	588030	+7.45

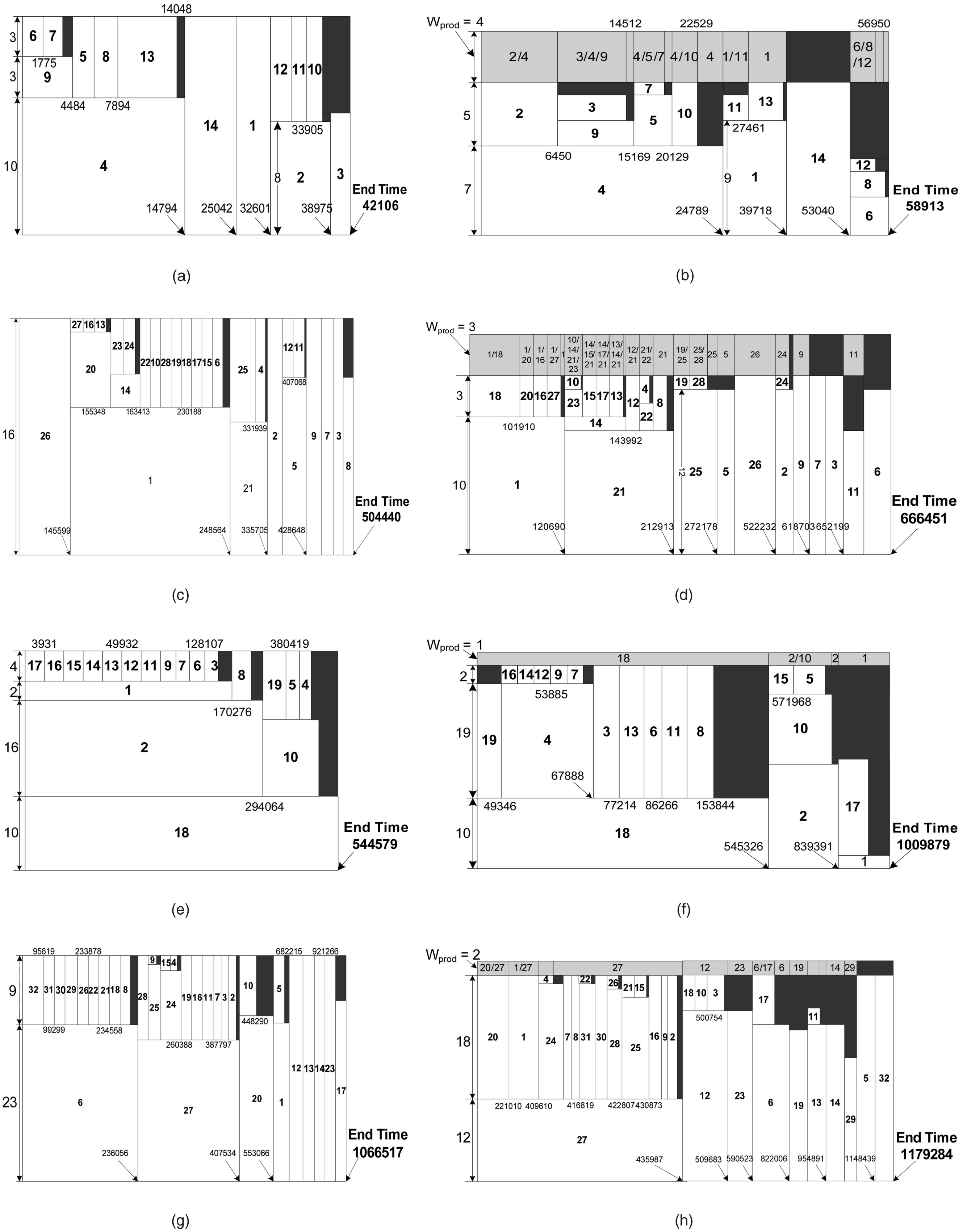


Fig. 15. Test schedule comparison of the two two-pattern test methodologies for benchmark SOCs (figures not drawn to scale). (a) g1023 schedule with enhanced WICs. (b) g1023 TPTEST schedule. (c) p22810 schedule with enhanced WICs. (d) p22810 TPTEST schedule. (e) p34392 schedule with enhanced WICs. (f) p34392 TPTEST schedule. (g) p93791 schedule with enhanced WICs. (h) p93791 TPTEST schedule.

TAM width of 16, producers' outputs are loaded from three TAM lines. Because many two-pattern tested cores are scheduled to be tested concurrently, the load size for those cores may increase. The TAT increases in this case by approximately 32 percent. For SOC p34392, only one TAM line is used to load producers' outputs for a total TAM width of 32 to get the optimal TAT in TPTEST mode. Although there are only four cores in TPTEST mode in this SOC, the TAT increases by 85 percent (from 544,579 to 1,009,879). This is because these four cores are the largest cores inside the SOC and the time spent on testing them dominates the overall TAT of the SOC. For this specific functional interconnect in our experiment, $Core_1$ and $Core_2$, and $Core_{10}$ and $Core_{18}$ are incompatible, from Fig. 15f, a good part of the testing time (from 153,844 to 545,326) is wasted in TPTEST mode, which is used to test in INTEST mode, however (see Fig. 15e). As shown in Section 6.3, the savings in DFT area for p34392 are not significant and, as a result, the system integrator may prefer to use the enhanced WIC design for p34392 to decrease SOC TAT. Nevertheless, unlike p34392, for SOC p93791, the sizes of the cores are similar and, consequently, none of the cores will dominate the whole SOC TAT. As a result, although test conflicts exist between cores, the size of idle rectangles is not too large (Fig. 15h). In this case, two of 32 TAM lines are used to transfer producers' outputs and TAT increases by only 12 percent.

7 Conclusion

Motivated by the difficulty in applying delay fault tests to embedded cores in broadside testing, this paper has presented a new test architecture for SOCs containing fully wrapped cores that need to be two-pattern tested. It was shown how IEEE Std. 1500 can be extended with a LOADPROD instruction for producer cores and a TPTEST instruction for CUT cores which ensure full controllability of the two-pattern tested core's primary inputs in two consecutive cycles. Solutions to address TAM design and test scheduling have also been elaborated. When compared to the case where enhanced WICs are used for two-pattern test, it was demonstrated that the proposed architecture can deliver the same test quality with less DFT area overhead and limited SOC TAT penalty.

Acknowledgments

A preliminary version of this paper was published in the *Proceedings of the IEEE/ACM Design Automation and Test in Europe Conference (DATE)*, pp. 744-749, March 2003.

References

[1] J. Aerts and E.J. Marinissen, "Scan Chain Design for Test Time Reduction in Core-Based ICs," *Proc. IEEE Int'l Test Conf. (ITC)*, pp. 448-457, Oct. 1998.

[2] K. Arabi, H. Ihs, C. Dufaza, and B. Kaminska, "Digital Oscillation-Test Method for Delay and Stuck-At Fault Testing of Digital Circuits," *Proc. IEEE Int'l Test Conf. (ITC)*, pp. 91-100, 1998.

[3] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," *Proc. Int'l Symp. Circuits and Systems (ISCAS)*, pp. 1929-1934, 1989.

[4] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing.* Kluwer Academic, 2000.

[5] K. Chakrabarty, R. Mukherjee, and A.S. Exnicios, "Synthesis of Transparent Circuits for Hierarchical and System-on-a-Chip Test," *Proc. IEEE Int'l Conf. VLSI Design (ICVD)*, pp. 431-436, Jan. 2001.

[6] P. Franco, S. Ma, J. Chang, C. Yi-Chin, S. Wattal, E. McCluskey, R. Strokes, and W. Farwell, "Analysis and Detection of Timing Failures in an Experimental Test Chip," *Proc. IEEE Int'l Test Conf. (ITC)*, pp. 691-700, 1996.

[7] J. Gatej, S. Lee, C. Pyron, and R. Raina, "Evaluating ATE Features in Terms of Test Escape Rates and Other Cost of Test Culprits," *Proc. IEEE Int'l Test Conf. (ITC)*, pp. 1040-1049, Oct. 2002.

[8] I. Ghosh, S. Dey, and N.K. Jha, "A Fast and Low-Cost Testing Technique for Core-Based System-Chips," *IEEE Trans. Computer-Aided Design*, vol. 19, no. 8, p. 863, Aug. 2000.

[9] S.K. Goel and E.J. Marinissen, "Effective and Efficient Test Architecture Design for SOCs," *Proc. IEEE Int'l Test Conf. (ITC)*, pp. 529-538, Oct. 2002.

[10] S.K. Goel and E.J. Marinissen, "Control-Aware Test Architecture Design for Modular SOC Testing," *Proc. IEEE European Test Workshop (ETW)*, pp. 57-62, May 2003.

[11] S.K. Goel and E.J. Marinissen, "Layout-Driven SOC Test Architecture Design for Test Time and Wire Length Minimization," *Proc. Design, Automation, and Test in Europe (DATE)*, pp. 738-743, Mar. 2003.

[12] I. Hamzaoglu and J.H. Patel, "Compact Two-Pattern Test Set Generation for Combinational and Full Scan Circuits," *Proc. IEEE Int'l Test Conf. (ITC)*, pp. 944-953, 1998.

[13] Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, O. Samman, Y. Zaidan, and S.M. Reddy, "Resource Allocation and Test Scheduling for Concurrent Test of Core-Based SOC Design," *Proc. IEEE Asian Test Symp. (ATS)*, pp. 265-270, Nov. 2001.

[14] IEEE Std 1500, *IEEE Standard for Embedded Core Test—IEEE Std. 1500-2004.* New York: IEEE, 2004.

[15] V. Immaneni and S. Raman, "Direct Access Test Scheme—Design of Block and Core Cells for Embedded ASICs," *Proc. IEEE Int'l Test Conf. (ITC)*, pp. 488-492, Sept. 1990.

[16] V. Iyengar and K. Chakrabarty, "Precedence-Based, Preemptive, and Power-Constrained Test Scheduling for System-on-a-Chip," *Proc. IEEE VLSI Test Symp. (VTS)*, pp. 368-374, May 2001.

[17] 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.

[18] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "Efficient Wrapper/TAM Co-Optimization for Large SOCs," *Proc. Design, Automation, and Test in Europe (DATE)*, pp. 491-498, Mar. 2002.

[19] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "Integrated Wrapper/TAM Co-Optimization, Constraint-Driven Test Scheduling, and Tester Data Volume Reduction for SOCs," *Proc. ACM/IEEE Design Automation Conf. (DAC)*, pp. 685-690, June 2002.

[20] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "On Using Rectangle Packing for SOC Wrapper/TAM Co-Optimization," *Proc. IEEE VLSI Test Symp. (VTS)*, pp. 253-258, Apr. 2002.

[21] V. Iyengar, S.K. Goel, K. Chakrabarty, and E.J. Marinissen, "Test Resource Optimization for Multi-Site Testing of SOCs under ATE Memory Depth Constraints," *Proc. IEEE Int'l Test Conf. (ITC)*, pp. 1159-1168, Oct. 2002.

[22] R. Kapur and T.W. Williams, "Manufacturing Test of SoCs," *Proc. IEEE Asian Test Symp. (ATS)*, pp. 317-319, Nov. 2002.

[23] S. Koranne, "Formulating SoC Test Scheduling as a Network Transportation Problem," *IEEE Trans. Computer-Aided Design*, vol. 21, no. 12, pp. 1517-1525, Dec. 2002.

[24] X. Lin, R. Press, J. Rajski, P. Reuter, T. Rinderknecht, B. Swanson, and N. Tamarapalli, "High-Frequency, At-Speed Scan Testing," *IEEE Design and Test of Computers*, vol. 20, no. 5, pp. 17-25, Oct. 2003.

[25] R. Madge, B.R. Benware, and W.R. Daasch, "Obtaining High Defect Coverage for Frequency-Dependent Defects in Complex ASICs," *IEEE Design and Test of Computers*, vol. 20, no. 5, pp. 46-53, Oct. 2003.

[26] E.J. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousberg, and C. Wouters, "A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores," *Proc. IEEE Int'l Test Conf. (ITC)*, pp. 284-293, Oct. 1998.

[27] E.J. Marinissen, S.K. Goel, and M. Lousberg, "Wrapper Design for Embedded Core Test," *Proc. IEEE Int'l Test Conf. (ITC)*, pp. 911-920, Oct. 2000.

[28] E.J. Marinissen, V. Iyengar, and K. Chakrabarty, "ITC'02 SOC Test Benchmarks Web Site," http://www.hitech-projects.com/itc02 socbenchm/, 2006.

[29] E.J. Marinissen, R. Kapur, M. Lousberg, T. Mclaurin, M. Ricchetti, and Y. Zorian, "On IEEE P1500's Standard for Embedded Core Test," *J. Electronic Testing: Theory and Applications*, vol. 18, nos. 4/5, pp. 365-383, Aug. 2002.

[30] P. Maxwell, R. Aitken, K. Kollitz, and A. Brown, "IDDQ and AC Scan: The War against Unmodelled Defects," *Proc. IEEE Int'l Test Conf. (ITC)*, pp. 250-258, 1996.

[31] M. Nourani and C. Papachristou, "Structural Fault Testing of Embedded Cores Using Pipelining," *J. Electronic Testing: Theory and Applications*, vol. 15, no. 1, p. 129, 1999.

[32] S. Pateras, "Achieving At-Speed Stuctural Test," *IEEE Design and Test of Computers*, vol. 20, no. 5, pp. 26-33, Oct. 2003.

[33] J. Rearick, "Too Much Delay Fault Coverage Is a Bad Thing," *Proc. IEEE Int'l Test Conf. (ITC)*, pp. 624-633, Nov. 2001.

[34] R.L. Wadsack, "Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits," *Bell Systems Technical J.*, pp. 1449-1474, May-June 1978.

[35] N. Touba and B. Pouya, "Using Partial Isolation Rings to Test Core-Based Designs," *IEEE Design and Test of Computers*, vol. 14, no. 4, pp. 52-59, Dec. 1997.

[36] P. Varma, "On Path Delay Testing in a Standard Scan Environment," *Proc. IEEE Int'l Test Conf. (ITC)*, pp. 164-173, Oct. 1994.

[37] P. Varma and S. Bhatia, "A Structured Test Re-Use Methodology for Core-Based System Chips," *Proc. IEEE Int'l Test Conf. (ITC)*, pp. 294-302, Oct. 1998.

[38] H. Vermaak and H. Kerkhoff, "Enhanced P1500 Compliant Wrapper suitable for Delay Fault Testing of Embedded Cores," *Proc. IEEE European Test Workshop (ETW)*, pp. 121-126, May 2003.

[39] L. Whetsel, "An IEEE 1149. 1 Based Test Access Architecture for ICs with Embedded Cores," *Proc. IEEE Int'l Test Conf. (ITC)*, pp. 69-78, Nov. 1997.

[40] Q. Xu, "Updated ITC'02 Benchmark SOCs with Random Functional Interconnect Information," http://www.ece.mcmaster.ca/~nicola/cadt.html, 2006.

[41] Q. Xu and N. Nicolici, "Resource-Constrained System-on-a-Chip Test: A Survey," *IEE Proc. Computers and Digital Techniques*, vol. 152, no. 1, pp. 67-81, Jan. 2005.

[42] T. Yoneda and H. Fujiwara, "Design for Consecutive Testability of System-on-a-Chip with Built-In Self Testable Cores," *J. Electronic Testing: Theory and Applications*, vol. 18, nos. 4/5, pp. 487-501, Aug. 2002.

[43] Y. Zorian, E.J. Marinissen, and S. Dey, "Testing Embedded-Core-Based System Chips," *Computer*, vol. 32, no. 6, pp. 52-60, June 1999.

[44] W. Zou, S.M. Reddy, I. Pomeranz, and Y. Huang, "SOC Test Scheduling Using Simulated Annealing," *Proc. IEEE VLSI Test Symp. (VTS)*, pp. 325-330, Apr. 2003.

Qiang Xu (S'03-M'06) received the BE and ME degrees in telecommunication engineering from Beijing University of Posts & Telecommunications, China, in 1997 and 2000, respectively. After working at a start-up integrated circuit design house for one and a half years, he continued his graduate study and received the PhD degree in electrical and computer engineering from McMaster University, Canada, in 2005. He is an assistant professor of computer science and engineering at The Chinese University of Hong Kong. His research interests lie in the broad area of computer-aided design with special emphasis on test and debug of system-on-a-chip integrated circuits. He received the Best Paper Award for the 2004 IEEE/ACM Design, Automation and Test in Europe (DATE) Conference and Exhibition. He is a member of the IEEE.

Nicola Nicolici (S'99-M'00) received the Dipl. Ing. degree in computer engineering from the University of Timisoara, Romania (1997), and the PhD degree in electronics and computer science from the University of Southampton, United Kingdom (2000). He is an assistant professor of computer engineering at McMaster University, Canada. His research interests are in the area of computer-aided design and test. He has authored a number of papers in this area and received the IEEE TTTC Beausang Award for the Best Student Paper at the International Test Conference (ITC 2000) and the Best Paper Award at the IEEE/ACM Design, Automation and Test in Europe Conference (DATE 2004). He is a member of the ACM SIGDA, the IEEE, and the IEEE Computer and Circuits and Systems Societies, and he serves on the editorial board of *IEE Proceedings—Computers and Digital Techniques*.