Modular SOC Testing With Reduced Wrapper Count

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

***Abstract*—Motivated by the increasing design for test (DFT) area overhead and potential performance degradation caused by wrapping all the embedded cores for modular system-on-a-chip (SOC) testing, this paper proposes a solution for reducing the number of wrapper boundary register (WBR) cells. By utilizing the functional interconnect topology and the WBRs of the surrounding cores to transfer test stimuli and responses, the WBRs of some cores can be removed without affecting the testability of the SOC. We denote the cores without WBRs as light-wrapped cores and present a new modular SOC test architecture for concurrently testing both the wrapped and the light-wrapped logic cores. Since the WBRs of cores that transfer test stimuli and test responses for light-wrapped cores become shared resources during test, conflicts arise during test scheduling that will negatively impact the test application time. As a consequence, to alleviate this problem, we present a novel test access mechanism (TAM) design algorithm for the proposed SOC test architecture. We conduct experiments on several SOC benchmark circuits and demonstrate that, with an acceptable increase in test application time, the number of WBRs can be significantly decreased. This will ultimately lessen the necessary DFT area for modular SOC testing and reduce the propagation delays between cores.**

***Index Terms*—Core wrapper, electronic test, system-on-a-chip (SOC).**

I. Introduction

SYSTEM-ON-A-CHIP (SOC) design using reusable intellectual property (IP) cores has become a state-of-the-art implementation paradigm that has triggered novel business models based on IP core providers and system integrators [7]. The IP cores are predesigned and preverified by the core providers; however, SOC composition is the system integrator's duty, who is also in charge of verification and manufacturing test of the entire SOC, including the IP-protected internal cores. Although the IP core reuse reduces the design cycle, the rapid increase in SOC complexity makes the test development a major implementation bottleneck [29]. This bottleneck is caused by the increasing number of internal cores, which cannot be tested easily since they are not directly accessible from the primary inputs (PIs) and primary outputs (POs). Various solutions for exploiting the SOC's architecture-specific information and using functional interconnect as test access mechanisms (TAMs), either at the core or at the system level, have been proposed [2], [3], [15], [21], [22], [28]. Regardless of their potential benefits in the long term, unless implemented automatically using a reliable test tool flow, these architecture-specific design for test (DFT) methodologies do not provide reusability, scalability, interoperability, and may become the computational bottleneck in the test automation flow. This problem is overcome by modular test strategies [29], which use dedicated bus-based TAMs for test data transportation. However, to enable both core reuse and easy test access, the embedded cores are connected to TAMs using special interfaces called core wrappers. Therefore, when the number of cores or the number of the core's terminals increases, the area introduced by core wrappers will also grow, which in turn adds to the overall cost of test. To address this issue, the main objective of this paper is to investigate how can the wrapper count be reduced while maintaining the benefits of modular SOC testing.

The rest of the paper is organized as follows. Section II reviews related work on modular SOC testing and motivates the research presented in this paper. Section III introduces light-wrapped cores that can facilitate a decrease in test resource usage necessary for modular SOC testing. In Section IV, we present a novel SOC test architecture with reduced wrapper count and provide the corresponding wrapper/TAM cooptimization algorithms. Section V describes our experiments and Section VI concludes this paper.

II. Prior Work and Motivation

This section overviews prior work on wrapper design and test architectures, and motivates the proposed research work.

A. Embedded Core Wrapper Design

An overview of the standard IEEE P1500 wrapper is shown in Fig. 1. Its main purpose is core isolation during test and it has three main modes of operation [20]: 1) functional operation, in which the wrapper is transparent; 2) inward-facing test mode (INTEST), in which test access is provided to the core itself; and 3) outward-facing test mode (EXTEST), in which test access is provided to the circuitry outside the core. The wrapper has a mandatory 1-bit input/output pair, wrapper serial input (WSI) and wrapper serial output (WSO), and optionally one or more multibit input/output pairs, wrapper parallel input (WPI) and wrapper parallel output (WPO). The wrapper also comprises wrapper boundary register (WBR) cells to provide controllability and observability for the core terminals and wrapper bypass register (WBY) cells to serve as a bypass for the test data access mechanism. In addition, the wrapper has a wrapper serial control (WSC) port and an internal wrapper instruction register (WIR) used to control the different operational modes of the wrapper. It is important to note that IEEE P1500 standard for embedded core test standardizes only the

Manuscript received August 31, 2004; revised December 14, 2004. An earlier version of this paper was presented at the 2003 IEEE International Test Conference (ITC), Charlotte, NC, September 28–October 3. This paper was recommended by Associate Editor K. Chakrabarty.

Q. Xu is with the Department of Computer Science and Engineering, Chinese University of Hong Kong, 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).

Digital Object Identifier 10.1109/TCAD.2005.852447

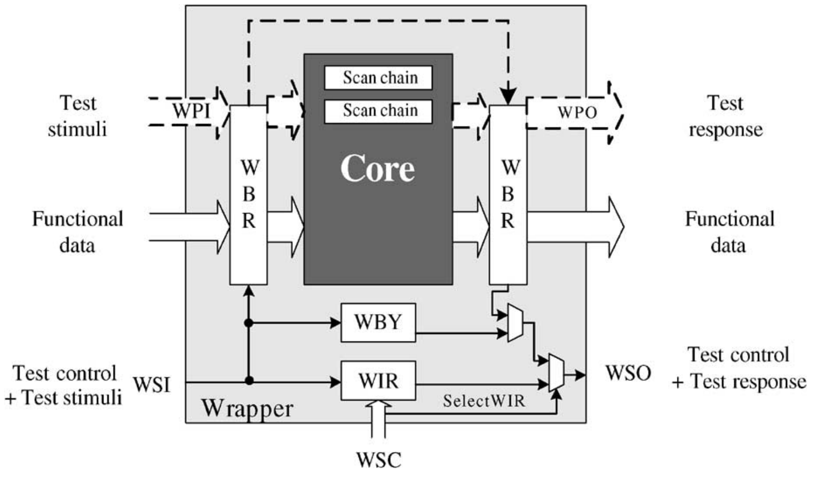

Fig. 1. Overview of IEEE P1500's wrapper architecture [20].

wrapper interface. Hence, the internal structure of the wrapper can be adapted to the specific SOC requirements.

Marinissen *et al.* [16] described a scalable core wrapper called TestShell, which forms the basis for the IEEE P1500 core wrapper [20]. The interconnection of internal scan chains and wrapper cells to the external TAM lines determines the construction of wrapper scan chains (WSCs). Since the test application time (TAT) of a core is dependent on the maximum WSC length, the main objective in wrapper optimization is to build balanced WSCs. Marinissen *et al.* [17] addressed this problem by describing a COMBINE heuristic for hard cores. Later, Iyengar *et al.* [9] 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., TAT varies with TAM width as a "staircase" function. According to this feature, only a few TAM widths between 1 and $W_{\max}$ (the maximum number of TAM width) are relevant when assigning TAM resources to hard cores, and these discrete widths are called pareto-optimal TAM widths.

B. SOC Test Architectures

Three basic types of SOC test architectures have been described in [1]: 1) the multiplexing architecture; 2) the daisy chain architecture; and 3) the distribution architecture. In the multiplexing and the daisy chain architecture, all cores get full access to all TAMs, while in the distribution architecture the total TAM is distributed over all cores. Two popular architectures that support more flexible test schedules are proposed based on the above architectures: The Test Bus architecture proposed in [24] can be seen as a combination of the multiplexing and the distribution architecture. While the TestRail architecture proposed in [16] is a combination of the daisy chain and the distribution architecture. Based on the TAM lines assignment strategy, the above modular test architectures can be further categorized into fixed-width test architectures and flexible-width test architectures. A vast body of research has been carried out for both types of architectures, and only a few representative approaches are summarized next.

For fixed-width test architectures, Iyengar *et al.* [9] 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 [10]. Koranne [14] formulated the test scheduling problem as a network transportation problem and presented a two-approximation algorithm to solve this problem. While the above approaches concentrate on Test Bus architecture, [4] presented an efficient heuristic TR-Architect that works for both Test Bus and TestRail architectures. In [5] and [6], TR-Architect was extended to account for the wire length cost and test control, respectively.

For flexible-width test architectures, Huang *et al.* [8] 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 algorithm to solve it. Iyengar *et al.* [12] presented an improved heuristic for the rectangle packing problem, when cores are supplied with fixed-length scan chains. Next in [11], the same authors extended their algorithm to incorporate precedence and power constraints while allowing a group of tests to be preemptable, while in [13], they considered minimizing the tester buffer reloads and multisite testing. Zou *et al.* [30] 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 an optimal test schedule.

C. Motivation and Summary of Contributions

The previously mentioned solutions for the design of core wrappers and test architectures [1], [4]–[6], [8]–[14], [16], [17], [24], [30] assume that all the cores attached to the TAM wires are fully wrapped, i.e., WBRs are placed on all the functional input and output terminals. While this guarantees core isolation during test, and hence high test quality, some embedded cores may have high pin count, and consequently the DFT area overhead associated with the wrappers will increase the cost of the test. Moreover, since both core's inputs and outputs

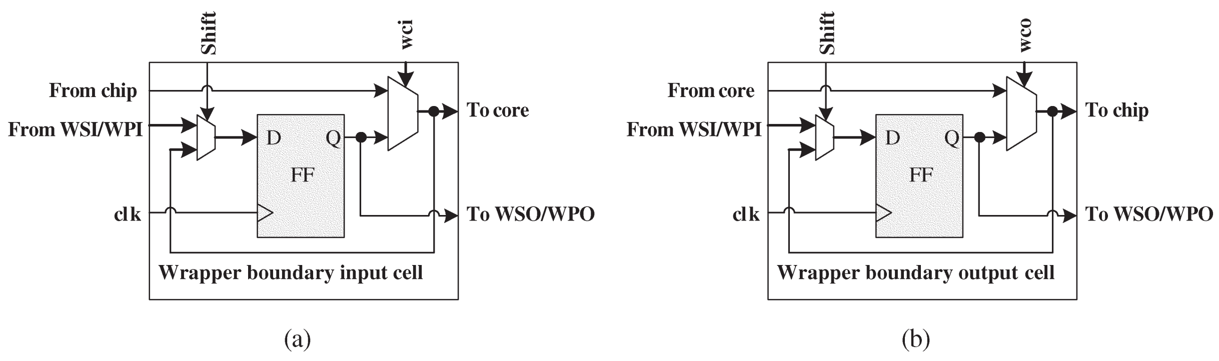

Fig. 2. Wrapper boundary cell for (a) core input and (b) core output terminal [20].

are buffered in the wrapper, at least two sets of multiplexers (Fig. 2 shows simple implementation examples of P1500 wrapper boundary cells) are required to switch between the functional and test modes of operation. If placed on the critical paths, these multiplexers will lower the maximum operating frequency, thus having a direct impact on the SOC's performance. An emerging challenge is to find ways of avoiding the performance penalty without affecting the test quality. To solve this problem, an approach based on partial isolation rings was proposed in [23]. Despite avoiding the high number of multiplexers, the main limitation of this solution lies in its computational complexity. This is because prior to deciding which input/output wrapper cells need to be inserted or removed, an analysis needs to be performed to check whether each test vector can be functionally justified. Therefore, the extensive usage of automatic test pattern generation (ATPG) for this analysis will reduce the methodology's scalability and reusability. Furthermore, the dependence of the wrapper cell removal methodology on the test set at hand will also limit the applicability of additional diagnosis data since the inserted DFT hardware will support only the preanalyzed test set.

To lower the DFT area and performance overhead by reducing the number of WBR cells, we believe the main challenge lies in finding a solution that will not only maintain the test quality but also be compatible with the IEEE P1500 standard [20] and, at the same time, preserve the modularity and scalability of the existing tool flows for TAM design and test scheduling. Consequently, the aim of this paper is to investigate the suitability of reusing the functional interconnect for transferring test data to cores whose input and output WBR cells are removed. The main contributions of this paper can be summarized as follows:

- a new concept called light-wrapped core is introduced to reduce the number of wrapper cells in the SOC without impacting its testability; experimental results on benchmark SOCs show that up to half of the cores can be unwrapped (the numbers are dependent on the interconnect topology) without affecting the test quality;
- since removing the wrapper cells from light-wrapped cores implies the usage of functional interconnects to transfer test data, we provide a detailed analysis of the new test conflicts that must be avoided in order not to compromise the test quality; based on this analysis, we introduce a new modular SOC test architecture that employs three separate TAM groups and facilitates concurrent testing of both P1500-wrapped cores and light-wrapped cores;
- we introduce new algorithms for wrapper/TAM cooptimization and test scheduling that fully exploit the proposed SOC test architecture; using experimental results on benchmark SOCs, we aim to improve the understanding of the limits of the inherent tradeoffs between the amount of DFT hardware and test application time required for modular SOC testing; it is essential to note that there is no loss in fault coverage for user-defined logic (UDL) since it can be modeled as a light-wrapped core, and using the proposed SOC test architecture and its associated algorithms UDL can be tested faster when compared to using serial EXTEST for it.

III. Light Wrappers for Embedded Cores

This section introduces a new concept called light wrapper and explains how light wrappers can be exploited during an SOC test to reduce WBR cells while maintaining the controllability/observability of the cores under test.

From the system integrator's standpoint, to test the embedded cores and their interconnects, full controllability and observability need to be provided at the inputs and outputs of each core. To ensure the modularity and scalability of an SOC test methodology, the controllability and observability of each embedded core should be test set independent. To achieve this, it is not necessary to wrap all the core's terminals with WBR cells, since the system integrator can also exploit the functional interconnect between cores to transfer the test data. To illustrate this observation, producers and consumers are introduced. For a given Core_i, its producers are the cores that feed its PIs and its consumers are the cores that capture its POs in the normal (functional) mode. Fig. 3 shows a part of an SOC, where Core_3 is not wrapped with WBR cells; however, all its producers $(\mathrm{Core}_1, \mathrm{Core}_2)$ and its consumer (Core_4) are P1500-wrapped. For INTEST of Core_3, the controllability of its input terminals is provided through its producers' output WBR cells while the observability of its output terminals is provided through its consumer's input WBR cells. In other words, we can shift in its test stimuli through the output WBR cells of Core_1 and Core_2, feed in the test stimuli into Core_3 through its normal functional path, and then capture its test response and shift it out through the input WBR cells of Core_4. Note, Core_3 cannot be tested using the EXTEST of Core_1,

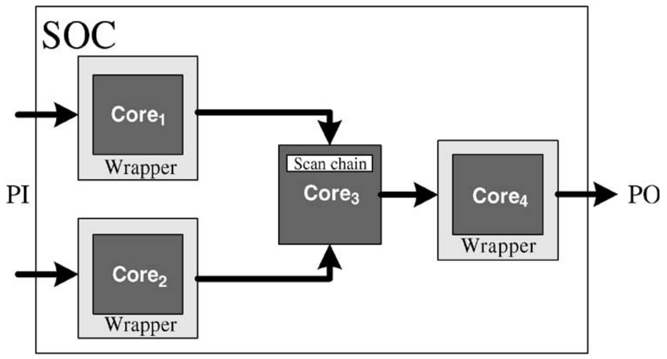

Fig. 3. Full controllability and observability for Core_3 without WBR cells.

Core_2, and Core_4 because the state of the internal scan chain in Core_3 cannot be controlled and observed in EXTEST mode. Since we apply test stimuli and capture test responses through functional paths, all the interconnects are tested implicitly, and hence we do not need to perform EXTEST for Core_3. It should be noted, however, that this implicit testing of interconnects loses the diagnostic information that differentiates between defects in Core_3's interconnects and defects in Core_3's internal logic. In this model, a P1500-wrapped core can serve as a producer and consumer at the same time because there are no test resource conflicts for using its wrapper output cells as a producer to shift in test stimuli and its wrapper input cells as a consumer to shift out test responses at the same time. Core_3 has a light wrapper [Fig. 4(a)] and the core wrappers for the producers and consumers need to be revised to support the previously explained transfer mechanisms [see Fig. 4(b) whose parameters are detailed in Section IV-B].

To summarize the above-explained observation, a core can be tested without wrapping its terminals as long as all its producers and consumers are P1500-wrapped. If the core does not have other test modes except the INTEST and EXTEST modes, then it does not need a wrapper at all [Fig. 4(a)]; however, the producers and consumers must be updated with a P1500-compliant wrapper [Fig. 4(b)] to support the proposed test strategy. If the core has other test modes, for example, it contains RAM or ROM blocks and has an additional built-in self-test (BIST) mode to test these internal memories, then, in addition to updating the producers' and consumers' wrappers, the core under test (CUT) needs a light wrapper without WBRs to support these additional modes, i.e., the light wrapper must include WIR and the WSC port to control the operational mode of the core. From now onwards, light-wrapped cores will refer to cores that do not need a wrapper at all or cores with a light wrapper, since both of them remove all the wrapper cells, which in turn reduce the DFT area and may improve the SOC's performance. The light-wrapped core requires either WSI/WSO or WPI/WPO to shift in the test stimuli and shift out the test responses to and from its internal scan chains. It may also include a serial or parallel bypass register (WBY) to enable a shortened test access path to other light-wrapped cores, if necessary. It is interesting to note that, if the light-wrapped core does not have internal scan chains, it can be treated as a UDL and the proposed test strategy for it is in essence a parallel EXTEST strategy.

In addition to the DFT hardware modification to support light-wrapped core testing, the P1500 instruction set also needs to be extended. New instruction LOADPROD for producer cores and UNLOADCONS for consumer cores are introduced. Moreover, if a core serves as both producer and consumer at the same time, an additional LDUNLDNEIGHBOR instruction is required to transfer test data both in and out of its WBR cells. These instructions are used to set the producer/consumer in the appropriate operational mode to shift in/out test stimuli/ responses.

IV. New SOC Test Architecture and Test Scheduling

Having introduced the light wrapper concept and outlined its applicability to P1500-based testing, this section focuses on its implications on SOC test architecture and test scheduling. Note, this paper does not address directly the design of hierarchical TAMs and the SOC hierarchy is assumed to be flattened. In addition, in this paper, we do not consider test scheduling constraints introduced by precedence relationship, preemption, and power. The introduction of the above features in the proposed SOC test architecture and test scheduling requires a separate investigation and consequently can be the topic of another study.

To clarify all the issues related to testing these light-wrapped cores, we provide a hypothetical SOC, called m4953, with nine cores and a system bus connecting three cores. The number of scan chains n_{sc} and the functional interconnects of these cores are shown in Fig. 5.[1] Note, the test infrastructure of the SOC has not been implemented yet and hence is not shown in the figure. Additional test parameters will be given in the experimental section. The name of this SOC follows the benchmark naming convention presented in [19], where m refers to McMaster University and the number 4953 denotes the test complexity.

A. Test Conflicts Caused by Sharing Functional Interconnect and Producers/Consumers

Before proposing a new SOC test architecture, we analyze the conflicts introduced by light wrappers. In the INTEST mode, all the P1500-wrapped cores can be tested concurrently as long as they use different TAM lines (assuming cores on the same TAM are tested in sequential order). However, because testing light-wrapped cores is dependent on their producers and consumers, TAM line conflicts are not the only ones that limit the test concurrency. Instead, there are five new types of test conflicts, described as follows.

Producer–CUT Conflict: Producer(s) and the CUT cannot be tested at the same time. For example, in Fig. 5, if Core_6 is a light-wrapped core, Core_2, Core_5, and Core_9 should not be tested at the same time as Core_6. 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.

[1]To make the drawing clear, the cores are not placed in increasing numerical order.

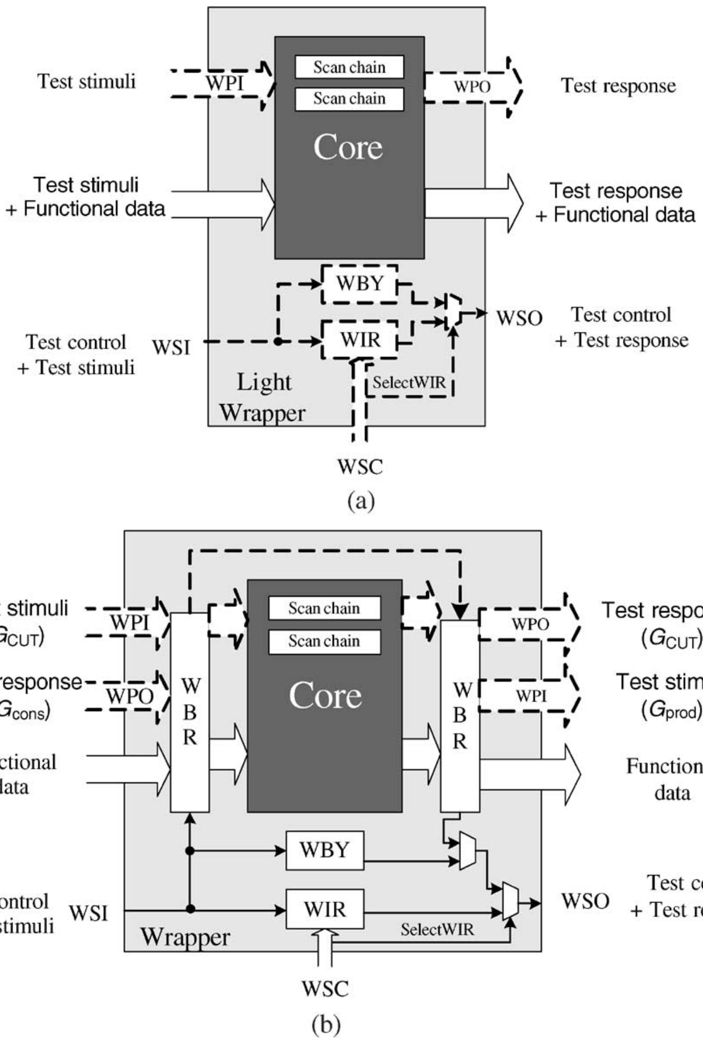
Test stimuli
WPI
Scan chain
Scan chain
Core
WPO
Test response
Test stimuli + Functional data
Test response + Functional data
WBY
WIR
WSO
Test control + Test response
Test control + Test stimuli
WSI
Light Wrapper
SelectWIR
WSC
(a)
Test stimuli (G_{CUT})
Test response (G_{cons})
Functional data
WBR
Test response (G_{CUT})
Test stimuli (G_{prod})
Wrapper
(b)

Fig. 4. Wrapper architectures for light-wrapped cores and their producer/consumer cores. (a) Light wrapper without WBR. (b) Revised IEEE P1500-compliant wrapper for producer/consumer cores.

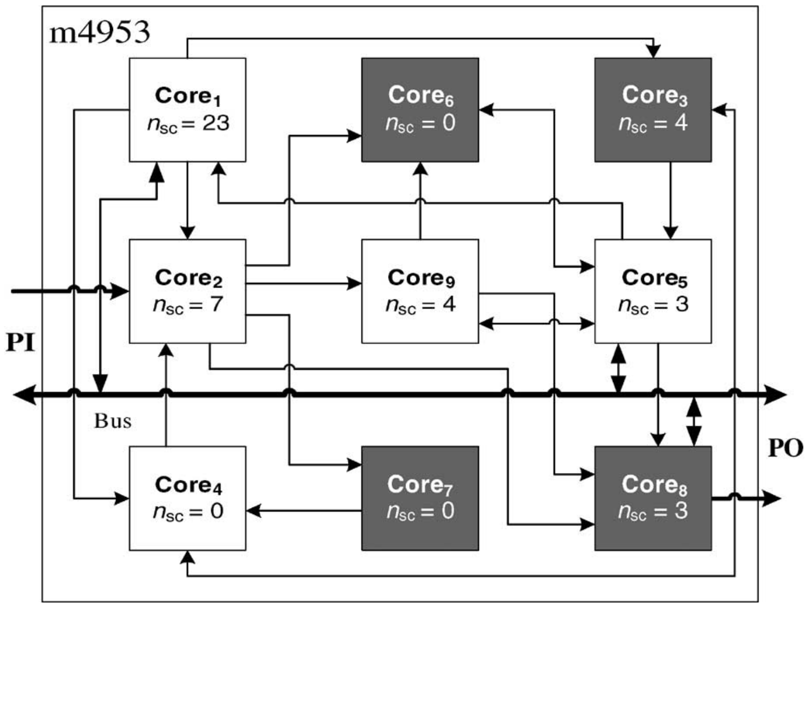
m4953
Core_1 $n_{sc} = 23$
Core_6 $n_{sc} = 0$
Core_3 $n_{sc} = 4$
Core_2 $n_{sc} = 7$
Core_9 $n_{sc} = 4$
Core_5 $n_{sc} = 3$
PI
Bus
PO
Core_4 $n_{sc} = 0$
Core_7 $n_{sc} = 0$
Core_8 $n_{sc} = 3$

Fig. 5. Example SOC: m4953.

CUT–Consumer Conflict: The CUT and consumer(s) cannot be tested at the same time. For example, in Fig. 5, if Core_2 is a light-wrapped core, Core_6, Core_7, Core_8, and Core_9 should not be tested at the same time. This is because the consumer needs to utilize its input WBRs to deliver test stimuli; however, at the same time, the CUT needs the consumer's input WBRs to capture the test responses. If they are tested concurrently, the test data will be corrupted.

Shared-Producer Conflict: Two light-wrapped cores that connect directly (i.e., on a dedicated nonshared set of lines) to the same producer cannot be tested at the same time. For example, in Fig. 5, if Core_7 and Core_8 are light-wrapped cores, they cannot be tested at the same time because both of them require the output WBRs of Core_2 to provide the test stimuli.

Shared-Consumer Conflict: Two light-wrapped cores that connect directly to the same consumer cannot be tested at the same time. For example, in Fig. 5, if Core_3 and Core_6 are light-wrapped cores, they cannot be tested at the same time because both of them need the input WBRs of Core_5 to capture the test responses.

Shared-Bus Conflict: If the producer(s) or consumer(s) connects to the light-wrapped CUT through functional buses, they can imply the previous described test conflicts and hence may not be tested at the same time. For example, in m4953, if Core_1 and Core_5 are two light-wrapped cores connected to the system bus, they cannot be tested at the same time because both of them need the I/O WBRs of Core_8 to provide test stimuli or capture the test response at the same time. However, if we have another wrapped core connected to the bus, for example, Core_4, then Core_1 and Core_5 can be tested together because we can use Core_4 as the producer and consumer of Core_1, and Core_8 as the producer and consumer of Core_5. By sharing the bus lines in consecutive times, there is only one clock cycle test application penalty per test pattern (using the same system bus to transfer test data), which is insignificant for scan-based testing.

B. *TAM Division Into Three Groups: Producers, CUTs, and Consumers*

The previous section has outlined the test conflicts that, if not taken into consideration, may corrupt the test data and render the test useless. Other types of conflicts may appear if the test data are transferred using shared TAM lines between producers, CUT, and consumers. To avoid this type of conflicts, which may adversely influence the overall TAT of the SOC, dividing the TAM lines into three groups is proposed, motivated by the following examples.

Example 1: Consider the SOC m4953 shown in Fig. 5 and let us assume that Core_1 and Core_2 are P1500-wrapped cores and Core_6 is a light-wrapped core, which needs Core_2 as a producer. We assume that Core_1 and Core_2 share the same TAM lines (TAM_{w1}) and Core_6 connects to a different TAM (TAM_{w2}). Since for testing Core_6 we need to use both TAM_{w1} and TAM_{w2} to transfer test data, loading a test pattern for Core_1 is prohibited while loading the stimulus for Core_6. As a result, there is a test conflict between Core_1 and Core_6 even though they connect to different TAM lines and have no functional relationship. This indirect TAM resource conflict may prohibit the overall test concurrency for light-wrapped cores in a large SOC, which will ultimately lead to testing all the light-wrapped cores separately, and thus resulting in very long testing time.

Sharing TAM lines between producers, CUTs, and consumers may also increase the test control complexity, as illustrated in the following example.

Example 2: In the case of m4953 shown in Fig. 5, if Core_2 is a light-wrapped core, then after the test stimuli are loaded in the output WBRs of Core_1 and Core_4, we must apply them at the same time. We also need to capture the test response in the input WBRs of Core_6, Core_7, Core_8, and Core_9 at the same time before shifting it out. If the TAM lines are shared between producers, CUTs, and consumers, all of these operations introduce additional synchronization issues and consequently they may increase not only the testing time but also the test control complexity.

If there are only a very small number of light-wrapped cores in the SOC, then using WSI/WSO to load/unload the test stimuli/responses may be a neat solution. After testing all the P1500-compliant cores, we can test these few light-wrapped cores one by one by putting its producers and consumers into the EXTEST mode and shift in/out its internal scan chains to/from WPI/WPO. However, if the number of light-wrapped cores is large, then the 1-bit TAM provides limited bandwidth for producers and consumers, and hence it will considerably increase the overall TAT of the SOC. When the number of light-wrapped cores is high, we propose a division of the TAM lines into three groups: G_{prod}, G_{CUT}, and G_{cons} used to load the producers, CUTs, and consumers, respectively. This division will remove the additional test conflicts discussed in Example 1 and test control complexity discussed in Example 2. Using the setup from Example 1, testing Core_6 will need the assistance of Core_2 to provide the test stimuli. If the output WBRs of Core_2 are loaded through G_{prod} and, although Core_1 and Core_2 share the same TAM resources in G_{CUT}, then Core_1 can still be tested at the same time as Core_6.

For the G_{CUT} group, we use a flexible-width test architecture, as introduced in Section II. For the G_{prod} and G_{cons} groups, however, we use the daisy chain architecture [1], i.e., long scan chains are constructed over all the producer cores' output terminals and all the consumer cores' input terminals, as depicted in Fig. 6. Producer bypass registers and consumer bypass registers (PBY and CBY in the figure) are introduced in order to shorten the loading/unloading time because only a few cores serve as producers or consumers at a specific test session. The main reason for using the daisy chain architecture for G_{prod} and G_{cons} group is to simplify the control complexity. When a producer (consumer) core is in LOADPROD (UNLOADCONS) mode, the producer (consumer) TAM lines go through the core's wrapper boundary cells, otherwise they go through the bypass register (note, it is unnecessary to introduce extra bypass instruction for producers and consumers). As a result, although testing light-wrapped cores involves several producers and consumers, they can be controlled by the LOADPROD and UNLOADCONS instructions independently. In addition, the daisy chain architecture for G_{prod} (G_{cons}) TAM groups can almost always give a near optimal loading

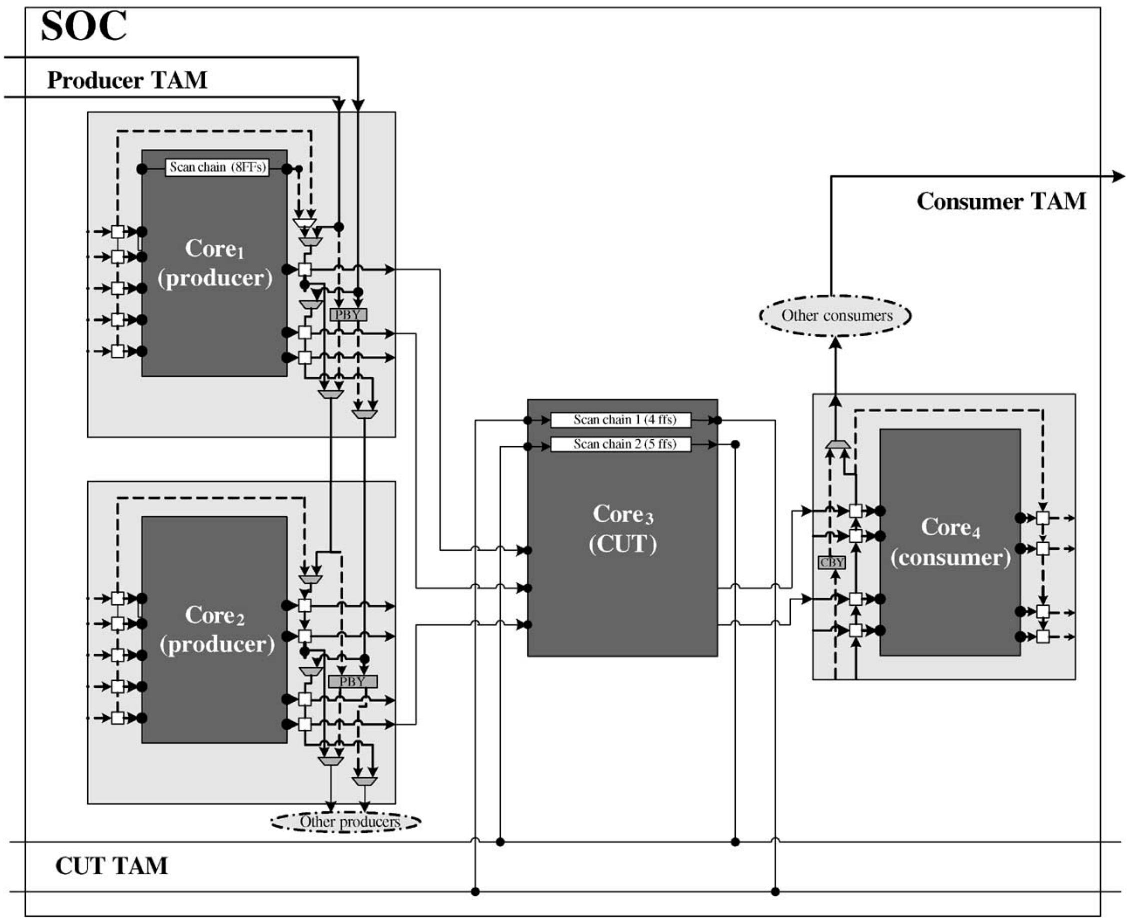

Fig. 6. Proposed test architecture for an example SOC containing light-wrapped cores.

(unloading) time for a given TAM width W_{prod} (W_{cons}). Suppose the number of the outputs of a producer is N_o, then its loading time will be $\lceil N_o/W_{\text{prod}} \rceil$. As long as $W_{\text{prod}} \leq N_o$ (which is realistic in most of the cases), there is no waste for G_{prod} TAM resources, except the few bypass cycles, which leads to a near optimal loading time for its producers. The same holds for G_{cons} unloading. It is essential to note that the TAT of a light-wrapped core is dependent on all the three TAM groups' architectures and the proposed TAM division into three groups facilitates concurrent testing of P1500-wrapped and light-wrapped cores, which is exploited by the algorithms described in the following section.

C. Proposed Algorithms for Wrapper/TAM Cooptimization

The introduction of light-wrapped cores, producers, consumers, and TAM division into three groups requires the development of new algorithms for wrapper/TAM cooptimization, as explained in this section. First we formulate the new problem to be solved.

Problem $P_{\text{LWT-opt}}$: Given the test set parameters for each core (including the number of PIs, POs, bidirectional I/Os, test patterns and scan chains, and each scan chain length), the total TAM width W_{ttl} for the SOC and the wrapper design constraints C_w determine the width of each TAM group (W_{prod}, W_{CUT}, and W_{cons} corresponding to G_{prod}, G_{CUT}, and G_{cons}), the TAM width and the wrapper design for each core, and a test schedule for the entire SOC such that 1) the wrapper design constraints C_w are satisfied; 2) the total number of light-wrapped cores is maximized; 3) the total number of TAM lines used at any time does not exceed W_{ttl}; and 4) the overall SOC TAT is minimized.

There are mainly three types of wrapper design constraints C_w. 1) If the critical paths appear between cores, then to avoid performance penalty some cores must be light wrapped. 2) If some of the cores are provided with P1500 wrappers and, due to their location and size, their overhead does not affect the performance or the cost of the SOC, then there is no reason to make them light wrapped. 3) If a core is two-pattern tested (e.g., targeting delay faults or CMOS stuck-open faults) as in [27], which employs the producers' WBR cells to apply the second consecutive pattern, double buffering in the WBRs of the core and its producers is necessary; hence, in this case, the CUT and all its producers must be P1500 wrapped. It is important to note that, if there are UDL blocks in the SOC, the system integrator has two choices: either make the UDL blocks P1500 wrapped and then use them as an input to the algorithms described in this section for problem $P_{\text{LWT-opt}}$ (this will guarantee that the maximum number of core wrappers is removed) or treat the UDL blocks as light-wrapped cores, i.e., they must satisfy the first wrapper design constraint when solving $P_{\text{LWT-opt}}$. In either case, there is no loss in fault

Algorithm 1 - TAM_Division_And_Schedule

INPUT: C_{set}, R, W_{ttl}, C_w, weight
OUTPUT: W_{prod}, W_{CUT}, W_{cons}, wrapper_type, schedule, $\mathrm{TAT}_{\mathrm{soc}}$

```
1. wrapper_type = Decide_Wrapper_Type ($C_{set}$, $R$, $C_w$);
2. TIG = Construct_TIG (wrapper_type, $R$);
3. For $W_{CUT}$ from $W_{ttl} - 2$ downto 1 {
4.    $W_{prod\_plus\_cons} = W_{ttl} - W_{CUT}$;
5.    For $W_{prod}$ from 1 to $W_{prod\_plus\_cons} - 1$ {
6.       $W_{cons} = W_{prod\_plus\_cons} - W_{prod}$;
7.       testing_time = Adapted_TAM_Schedule_Optimizer(
.                       $C_{set}$, TIG, $W_{prod}$, $W_{CUT}$, $W_{cons}$);
8.       localmin = min{all testing_time};
9.       Record $W_{prod\_localmin}$, $W_{cons\_localmin}$;
.     }
10.   if (localmin > weight × globalmin) {
11.      break; }          /*Prune search space*/
12.   globalmin = min{all localmin};
13.   Record $W_{prod\_globalmin}$, $W_{cons\_globalmin}$;
.  }
14.   $W_{prod} = W_{prod\_globalmin}$;
.     $W_{cons} = W_{cons\_globalmin}$;
.     $W_{CUT} = W_{ttl} - W_{prod} - W_{cons}$;
.     $TAT_{soc}$ = globalmin;
15. return $W_{prod}$, $W_{CUT}$, $W_{cons}$, wrapper_type, schedule, $TAT_{soc}$;
```

Fig. 7. Pseudocode for wrapper/TAM cooptimization for SOC with light-wrapped cores.

coverage of UDL since, by construction, it is ensured that each light-wrapped core is controlled by its producers and observed by its consumers. Therefore, the proposed solution can also be used as an alternative to EXTEST for concurrently testing wrapped cores and UDL.

In the rest of this section, we first present the top level algorithm for solving $P_{\mathrm{LWT-opt}}$ and then give details on the new procedures and concepts specific to our approach.

TAM Division and Test Scheduling: The proposed algorithm TAM_Division_And_Schedule to solve $P_{\mathrm{LWT-opt}}$ is shown in Fig. 7. The inputs are the set of cores (C_{set}), TAM width (W_{ttl}), functional interconnect relationship between cores (R), wrapper design constraint (C_w), and a weight parameter (weight), used in pruning the search space. The outputs are the number of TAM lines allocated to each TAM group, wrapper type (wrapper_type) and design for each core, SOC test schedule (schedule), and the overall test application time for the entire SOC $(\mathrm{TAT}_{\mathrm{soc}})$. The optimal TAM division, i.e., the combination of W_{prod}, W_{CUT}, and W_{cons} that gives the minimum TAT of the SOC, is acquired through enumeration. The enumerative algorithm begins by determining which cores must be light wrapped (line 1), according to the functional interconnect relationship R and predefined wrapper design constraint (C_w) of the SOC. Based on the generated wrapper type (light wrapped or not) for each core and the test conflicts determined by the functional interconnect relationship between cores (R), a test incompatibility graph (TIG) is created (line 2). Next, the algorithm will enumeratively find the optimal TAM division and the minimum system TAT $\mathrm{TAT}_{\mathrm{soc}}$. In the inner loop (lines 5 to 9), the local minimum TAT localmin for a fixed total width of $W_{\mathrm{prod}} + W_{\mathrm{cons}}$ $(W_{\mathrm{prod\_plus\_cons}})$ is computed. In the outer loop (lines 3 to 13), the algorithm searches for globalmin, among the localmin values, by enumerating W_{CUT} from the maximum possible value $W - 2$ to 1. During our initial experiments, it was observed that localmin is nearly a convex function with respect to W_{CUT}. That is, it keeps decreasing until it reaches a local minimum value, at which point it starts increasing. This convex attribute can be explained by the fact that when W_{CUT} has a small value, the TAT is dominated by the time to transfer test data through G_{CUT} [for justification, see (1) and (2) explained later in this section]. Increasing W_{CUT}, and hence decreasing $W_{\mathrm{prod}} + W_{\mathrm{cons}}$, will finally break this bottleneck. TAT starts to increase when the time required to load/unload the producer/consumer output/input WBR cells starts to dominate the scan time for G_{CUT}. There are some variations around the local minimum value, which can be justified by the heuristic nature of the dynamic rectangle packing algorithm (explained later in this section). Hence, to prune the search space, we enumerate the localmin values in the opposite direction (i.e., from $W - 2$ to 1), since we want to discard the large localmin values. To accommodate the variations around the minimum value, we use a parameter weight (a real value slightly greater than 1) (lines 10 and 11). It should be noted that during the enumeration process, we do not need to do TAM design for the G_{prod} and G_{cons} groups, since the implementation of the daisy chain architectures for these two groups is straightforward once W_{prod} and W_{cons} are determined. To generate a TAM design for G_{CUT}, we adapt an existing generalized rectangle packing algorithm TAM_Schedule_Optimizer [12]. Due to the usage of the daisy chain architecture for producers/consumers, a dynamic adaptation of this existing algorithm is necessary. We elaborate on each of the main steps of the top-level algorithm in the following paragraphs.

The worst case complexity of the algorithm TAM_Division_And_Schedule is $O(W_{\mathrm{ttl}}^2 \times \mathrm{C(ART)})$, where C(ART) is the worst case complexity of algorithm Adapted_TAM_Schedule_Optimizer, which will be detailed at the end of this section.

Decide Wrapper Type: Not all the cores need to be P1500 wrapped in an SOC; however, to provide full controllability and observability, each light-wrapped core needs to be surrounded by P1500-wrapped cores, i.e., all its producers and consumers must be wrapped. The pseudocode for deciding the wrapper type is shown in Fig. 8. The algorithm takes the set of cores C_{set}, the functional interconnect relationship R, and the wrapper design constraints C_w as the inputs, and it outputs the wrapper type for each core $i \in C_{\mathrm{set}}$. First, the cores that need to be wrapped by P1500-compliant wrappers according to the direct functional relationship (i.e., dedicated nonshared communication lines) are identified (lines 1 to 10). In the first loop (lines 1 to 6), we initialize the wrapper status and wrap the cores according to wrapper constraints, if any. For all the other cores, the wrapper is first set to a light-wrapped type and a variable called test_dependency is initialized to the sum of its unwrapped producers and consumers (note, if one core serves as both a producer and a consumer for another core, it is not to be counted twice). This variable is used to indicate the core's test requirements as a light-wrapped core; if this number is large, it means that when this core is light wrapped, we need

Algorithm 2 - Decide_Wrapper_Type

INPUT: C_{set}, R, C_w
OUTPUT: wrapper_type

```
/* According to direct functional interconnects */
1. For each Core $i \in C_{\text{set}}$ {
2.    if ($C_w$ exist) {
3.        Wrap core $i$ according to its constraint
4.    } else {
5.        Set Is_Light_Wrapped$_i$ = true;
6.        Initialize test_dependency$_i$;
.     }
. }
7. While (test_dependency$_i$ != 0 for any core $i \in C_{\text{set}}$) {
8.    Find Core $j$ with the maximum test_dependency;
9.    Set Is_Light_Wrapped$_j$ = false; test_dependency$_j$ = 0;
10.   Update test_dependency of its producers and consumers;
. }
/* According to functional bus interconnects */
11. For each functional bus
12.   if (No core on the bus is wrapped)
13.       Wrap the core with the least number of I/Os;
14. return wrapper_type for each core;
```

Fig. 8. Procedure for deciding the wrapper type of each core.

a large number of P1500-wrapped neighbor cores to test it. For example, in the case of m4953, Core$_2$ has two producers (Core$_1$ and Core$_4$) and four consumers (Core$_6$, Core$_7$, Core$_8$, and Core$_9$), hence its test_dependency is initialized to 6. If Core$_2$ is a light-wrapped core, we need to wrap all its six neighbors. As a result, it is better to wrap Core$_2$ with a P1500-compliant wrapper. Therefore, the algorithm finds the cores with a large test_dependency and wraps them as P1500-compliant (lines 8 and 9). Whenever a core is decided to be wrapped as P1500-compliant, its test_dependency is set to 0 because it does not require any other cores to facilitate its test; the test_dependency of all its light-wrapped producers/consumers is deducted by 1 (line 10). When functional busses are used, at least one core on each functional bus must be wrapped as P1500-compliant to test all the other light-wrapped cores on the bus (lines 11 to 13). The algorithm will find a core with the least number of inputs and outputs to wrap in order to decrease the time required to load/unload the test stimuli/responses. To illustrate the outcome of the proposed algorithm, in the case of m4953, Core$_3$, Core$_6$, Core$_7$, and Core$_8$ are selected to be light wrapped, as shown by the shaded boxes in Fig. 5.

Construct the TIG: If there are test conflicts between two cores (see Section IV-A), then these two core tests cannot be scheduled at the same time and are denoted as incompatible cores. We construct a TIG by treating each core as a node and adding an edge between two nodes if they are incompatible. This TIG is used in Algorithm 3 (Adapted_TAM_Schedule_Optimizer). The TIG generated for m4593 is shown in Fig. 9. As shown in the figure, edges illustrating incompatibility can exist only between two light-wrapped cores or between a light-wrapped core and its producers/consumers. Two P1500-wrapped cores are always compatible during test

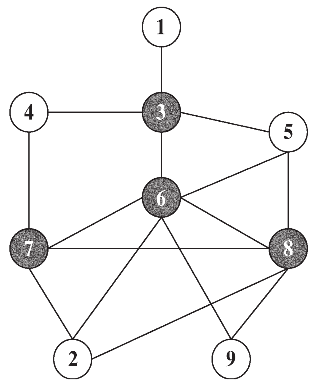

Fig. 9. m4953 TIG.

because they do not need each other's help to test their internal logic.

Dynamic Rectangle Representation: For a P1500-wrapped core, if the assigned TAM width in G_{CUT} is given, the TAT to apply the entire test set T_p is determined by (1) [17], 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 [9] for wrapper optimization to build balanced WSCs, $s_i(s_o)$ has a fixed value for a given TAM width, and hence the core test can be represented as a static rectangle

$$T_p = (1 + \max\{s_i, s_o\}) \times p + \min\{s_i, s_o\}. \qquad (1)$$

However, for a light-wrapped core, its TAT T_l does not only depend on the time to load/unload its own producers (L_{prod}), consumers (L_{cons}), and internal scan chains (L_{in}); T_l also depends on the time necessary to load/unload all the concurrently tested light-wrapped cores' producers/consumers. To keep the control and computational complexity low, we propose to align test patterns for all the concurrently tested light-wrapped cores and hence T_l is calculated as

$$T_l = \sum_s \left(1 + \max\left\{\sum L_{\text{prod}}, \sum L_{\text{cons}}, \max\{L_{\text{in}}\}\right\}\right) \times (p_s + 1) \qquad (2)$$

where bypass cycles are ignored.

As a result, if for a given light-wrapped 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 TAT will be computed based on the light-wrapped cores scheduled in each of these s divisions. The following example is used to better illustrate the computation of T_l.

Example 3: In the case of m4953, Core$_3$ is compatible with light-wrapped cores Core$_7$ and Core$_8$. Let us assume Core$_7$ and Core$_8$ are selected to be scheduled at the same test time with Core$_3$, as shown in Fig. 10 (the given number of test patterns for these three cores has been selected only to illustrate this example). The time necessary to apply a pattern for Core$_3$ is updated each time the schedule changes. For the first ten patterns, the shifting time for each pattern of both

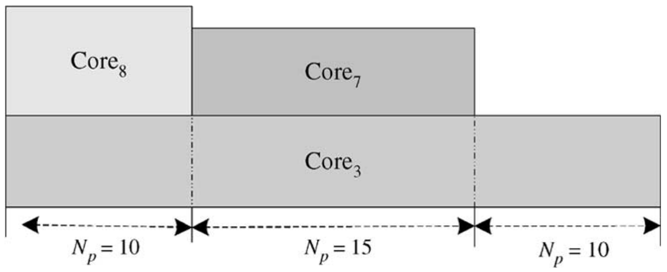

Fig. 10. Test application time for light-wrapped cores.

Core_3 and Core_8 will be $\max\{L_{\mathrm{prod\_3}} + L_{\mathrm{prod\_8}}, L_{\mathrm{cons\_3}} + L_{\mathrm{cons\_8}}, L_{\mathrm{in\_3}}, L_{\mathrm{in\_8}}\}$. However, for the next 15 patterns, once the test for Core_8 has been completed and Core_7 is scheduled concurrently with Core_3, the shifting time for each pattern will be $\max\{L_{\mathrm{prod\_3}} + L_{\mathrm{prod\_7}}, L_{\mathrm{cons\_3}} + L_{\mathrm{cons\_7}}, L_{\mathrm{in\_3}}, L_{\mathrm{in\_7}}\}$. The same reasoning is applied for the last ten patterns when Core_3 is not concurrent with any other light-wrapped cores. The shifting time for each pattern will be $\max\{L_{\mathrm{prod\_3}}, L_{\mathrm{cons\_3}}, L_{\mathrm{in\_3}}\}$. The variations in the shifting time for each of the three divisions of the schedule for Core_3 can be differentiated using a $\mathrm{loadSize} = \max\{\sum L_{\mathrm{prod}}, \sum L_{\mathrm{cons}}, \max\{L_{\mathrm{in}}\}\}$, determined by all the concurrently tested light-wrapped cores.

From the above discussion, we can see that the T_l for a light-wrapped core changes every time when the schedule is updated. This dynamic attribute leads to a dynamic rectangle representation of the light-wrapped core's test and is caused by the usage of the daisy chain architecture for producer/consumer TAM groups.

Adapted Dynamic Rectangle Packing: To concurrently test the light-wrapped and P1500-wrapped cores, for the CUT TAM group we use an Adapted_TAM_Schedule_Optimizer algorithm (the pseudocode is shown in Fig. 11). The algorithm takes the core list C_{set}, TIG, and the TAM division as inputs, and generates the schedule for each core and the overall TAT of the SOC. The algorithm is based on a generalized rectangle packing algorithm TAM_Schedule_Optimizer proposed in [12]. TAM_Schedule_Optimizer first finds out the pareto-optimal TAM widths for each core. Next, a "preferred TAM width" 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 [12] for more details on terminology. It should be noted that in the following we only emphasize the specifics of the proposed solution and the differences with respect to the original algorithm from [12].

As described earlier, for light-wrapped cores, the test cannot be precomputed and represented as a static rectangle; its TAT (the width of the rectangle) varies with its schedule,

Algorithm 3 - Adapted_TAM_Schedule_Optimizer

INPUT: C_{set}, TIG, W_{prod}, W_{CUT}, W_{cons}
OUTPUT: schedule, testing_time

```
1. Compute collection $R_p$ of rectangles for P1500-compliant core set $C_p$;
2. Initialize($C_p$, $d$, $p$);          /*$C_p$ is the P1500-compliant core set*/
3. Set $C_{\mathrm{unfinished}} = C_{\mathrm{set}}$; w_avail = $W_{\mathrm{CUT}}$; this_time = 0; (see[12])
4. While $C_{\mathrm{unfinished}} \neq \emptyset$ {
5.    if w_avail > 0 {
6.       Find unscheduled light-wrapped core set $C'_d$;
7.       Compute collection $R'_d$ of dynamic rectangles for $C'_d$;
8.       Initialize($C'_d$, $d$, $p$);
9.       Schedule compatible P1500-wrapped cores that can be assigned
.        preferred TAM width or compatible light-wrapped cores; (see [12])
10.      Schedule compatible P1500-wrapped cores that can use the resulting
.        idle TAM wires; (see [12])
11.      Update $L_{\mathrm{prod}}$, $L_{\mathrm{cons}}$, loadSize;
12.      Update test_time for scheduling light-wrapped cores;
13.   } else {
14.      Update next_time;
15.      Find unscheduled light-wrapped cores $C''_d$ with
.        no internal scan chains;
16.      Compute collection $R''_d$ of dynamic rectangles for $C''_d$
17.      Initialize($C''_d$, $d$, $p$);
18.      Schedule compatible cores in $C''$ not exceeding next_time;
19.      Update $L_{\mathrm{prod}}$, $L_{\mathrm{cons}}$, loadSize;
20.      Update test_time for scheduling light-wrapped cores;
21.      Update next_time;
22.      Finish the scheduling core test $C_i$ with ending time next_time;
23.      Update this_time;
24.      w_avail + = $w_{\mathrm{tam\_}C_i}$;
25.      $C_{\mathrm{unfinished}}$ − = $\{C_i\}$;
26.      Update this_time;
.     }
.  }
27. return schedule, testing_time;
```

Fig. 11. Procedure for test scheduling with given widths of each TAM group.

hence its rectangle representation is computed dynamically (lines 7 and 16). In addition, since the TAT of the light-wrapped cores may change dynamically with its schedule, there are no "preferred TAM widths" for them. In [12], the procedure Initialize (line 2) was used to compute the preferred width for each core; the parameters d and p were sometimes 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 $p = 1.0$ (these two values give a generally good "preferred TAM width"); this may result in a different schedule and a slightly longer TAT in some cases when compared to the result in [12]. Line 3 initializes the cores that have not finished their schedule $C_{\mathrm{unfinished}}$, the currently available TAM width w_avail, and the current start time for unscheduled cores this_time, respectively. In line 9, the algorithm tries to schedule either a P1500-wrapped core with preferred TAM width or a light-wrapped core with the maximum allowable test pattern count that is able to fit in the idle rectangle (since test patterns for concurrently tested light-wrapped cores are aligned). When scheduling a light-wrapped core, its rectangle size is determined as the following. The TAM width for this core (its height) is the minimum value that minimizes load-Size and the TAT of the core (its width) is calculated using (2). Once a core is scheduled (lines 9 and 10), the available CUT TAM width w_avail will be deducted the value of the assigned CUT TAM width for the core. Whenever a light-wrapped core is scheduled, L_{prod}, L_{cons}, and loadSize, for the currently scheduled light-wrapped cores, need to be updated (lines 11 and 19) and TAT will be recalculated (lines 12 and 20). If a light-wrapped core has no internal scan chains inside and hence does not need any TAM lines in the G_{CUT} group, we may be able to schedule it even when the available CUT TAM width w_avail $= 0$ in G_{CUT} (lines 15 to 20). This is because only G_{prod} and G_{cons} resources are necessary. Once there is no core able to be tested starting with this_time, the currently scheduled core with the minimum TAT will be finished (line 24), its TAM resources are released, and this_time advances to its finishing time; the algorithm try to schedule another core with these TAM resources. Note, due to test conflicts, we are only able to select a compatible core to be scheduled at any time (lines 9, 10, and 18); this is done through checking whether there is an edge in TIG between the cores currently under test and the to-be-scheduled core.

The worst case complexity C(ART) of the algorithm Adapted_TAM_Schedule_Optimizer can be estimated as follows. The while loop in line 4 of Fig. 11 is executed N_c times, where N_c is the number of cores of the SOC. In each such execution, a linear search in the $O(N_c)$ core set is used to find the next core to be scheduled. In addition, these cores are also examined $O(N_l)$ to determine whether they are compatible with the currently scheduled light-wrapped cores (lines 9, 10, and 15). Moreover, in each such execution, a collection of $O(W_{\mathrm{CUT}})$ rectangles is generated for $O(N_l)$ light-wrapped cores. The complexity of rectangle generation using Design_wrapper is $O(\mathrm{sc} \log \mathrm{sc} + \mathrm{sc} \cdot k)$, in which sc is the number of scan chains in the light-wrapped core and k is the TAM width [9]. As a result, the worst case complexity C(ART) is $O(N_c^2 \cdot N_l^2 \cdot W_{\mathrm{CUT}} \cdot (\mathrm{sc} \log \mathrm{sc} + \mathrm{sc} \cdot W_{\mathrm{CUT}}))$.

Summary: In this section, we have presented a new modular SOC test architecture with reduced wrapper count. We have described the algorithms used for concurrent test scheduling of both P1500-wrapped and light-wrapped cores and have analyzed their computational complexity.

V. Experimental Results

The purpose of our experiments is to find out how much test area can be saved, without affecting the test quality, and what are the implications of these savings on testing time. In addition to the hypothetical SOC m4953, benchmark SOCs from the ITC'02 SOC test benchmarking initiative ([18]) are used in our experiments. Since the functional interconnects are not provided in the original benchmark files, we have randomly generated them to support the proposed approach, including the direct connection between cores and functional busses [26]. Through randomization, we wanted to investigate what is the average impact of the proposed algorithms on the number of light-wrapped cores and the overall TAT. We have assumed that the SOCs have $N_c/10$ busses 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 a bus are assumed to be able to transfer data to and from the bus. We have also assumed that every core has a random number of $q (1 \leq q \leq 3)$ producers, while the consumers for each CUT are generated from the producer–CUT relationship.

TABLE I
TEST PARAMETERS OF SOC m4953

	N_{in}	N_{out}	N_{bi}	N_{sc}	SC_{length}
Core$_1$	35	50	28	23	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 4 4 4 4 4 4
Core$_2$	117	84	36	7	221 221 221 221 221 200 178
Core$_3$	144	47	72	4	149 149 149 147
Core$_4$	202	60	20	0	
Core$_5$	126	25	40	3	300 299 299
Core$_6$	29	40	6	0	
Core$_7$	6	242	0	0	
Core$_8$	136	12	28	3	160 159 159
Core$_9$	194	157	6	4	149 149 149 147

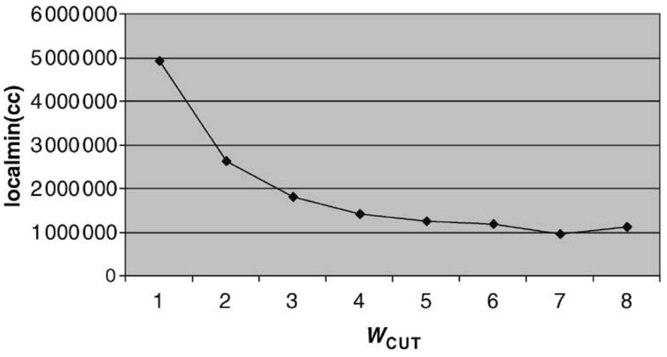

Fig. 12. Test application time variation with W_{CUT}.

It should be noted that there are no wrapper design constraints in our experiments and the proposed algorithm in Section IV determines the wrapper type of each core, the optimal TAM division, and the test schedule. We have divided our experiments into three subsections. First, we discuss the test schedule for hypothetical SOC m4953, then we analyze the number of cores that can be light wrapped, and finally we discuss the testing time implications of using the proposed TAM design algorithm for testing the light-wrapped cores.

A. Experiment 1: Test Schedule Comparison for SOC m4953

First, we investigate our pruning technique used for rapidly dividing the available TAM lines into three separate groups: producer, CUT, and consumer for SOC m4953. The test parameters for the cores in m4953 are shown in Table I, in which N_{in}, N_{out}, N_{bi}, and N_{sc} denote the number of inputs, outputs, bidirectionals, and scan chains in the specific core, respectively. The length of each scan chain is shown in column SC_{length}.

The TAT variation with W_{CUT} for m4953 is depicted in Fig. 12 (given the total TAM width $W_{\mathrm{ttl}} = 10$). Using the proposed TAM_Division_Schedule algorithm, we obtain the minimum TAT of 955911 clock cycles for $W_{\mathrm{CUT}} = 7$, $W_{\mathrm{prod}} = 2$, and $W_{\mathrm{cons}} = 1$. For this particular case, we deal with a convex function, and the first identified local minimum

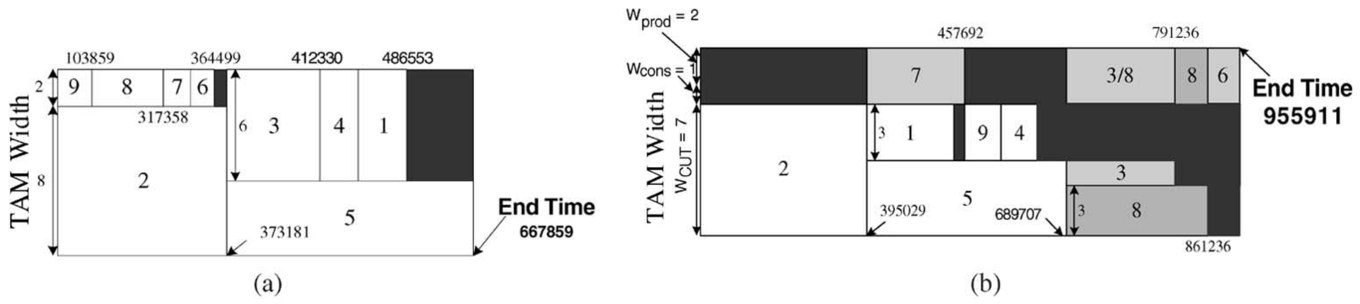

Fig. 13. Test schedule comparison for SOC m4953. (a) Test schedule when all the cores are P1500-wrapped using [12]. (b) Test schedule when Core_3, Core_6, Core_7, and Core_8 are light wrapped using the new method.

is the only global minimum. If we set $\text{weight} = 1.1$ (see Algorithm 1), the search for the minimum TAT will start from $W_{\text{CUT}} = 8$ and stop at $W_{\text{CUT}} = 6$. This will prune the search space and hence reduce the computational time to a few seconds even for large SOCs while getting the best possible TAM division and schedule.

In Fig. 13(a), we present the test schedule obtained for m4953 when all the cores are P1500-wrapped. When applying the new Decide_Wrapper_Type, if no wrapper design constraints exist, Core_3, Core_6, Core_7, and Core_8 are selected to be light wrapped, and the test schedule is shown in Fig. 13(b). We can observe that SOC TAT increases by approximately 45% due to the following three main reasons.

1) The test conflicts introduced in the light-wrapped SOC test model cause more idle rectangle areas in the bin. For example, although Core_6 can fit into the rectangle area above Core_2, due to the producer–CUT conflict, Core_6 is incompatible with Core_2 and hence they cannot be scheduled at the same time.
2) The number of TAM lines used to access the P1500-compliant cores and the internal scan chains of light-wrapped cores (G_{CUT}) are decreased from 10 to 7. To support light-wrapped core testing, two TAM lines are used for loading the producers' outputs and one TAM line is used to unload the consumers' inputs.
3) When two light-wrapped cores are scheduled at the same time, the load time for each overlapped pattern may increase. In this example, light-wrapped Core_3 and Core_8 are tested concurrently, and the load time of the overlapped test patterns is dominated by the loading time of the producer outputs.

It can be observed in Fig. 13(b) that Core_7 is scheduled from the same starting time as Core_1 even when the available number of CUT TAM lines $G_{\text{CUT}} = 0$. This is because Core_7 has no internal scan chains and test data can be transferred only using G_{prod} and G_{cons}.

B. Experiment 2: Number of Light Wrappers and Reduction in WBRs

Table II shows the reduction in the number of light-wrapped cores and the number of WBRs for the 100 random-generated interconnects for four benchmark SOCs [18]. N_c is the total number of cores while $N_{\max\_l}$, $N_{\min\_l}$, and N_{ave_l} denote the maximum, minimum, and average number of light-wrapped cores, respectively. N_{wbr} is the total number of WBRs and $N_{\text{max\_wbr}}$, $N_{\text{min\_wbr}}$, and $N_{\text{ave\_wbr}}$ denote the maximum, minimum, and average number of WBRs that are removed. The percentage reductions are defined as $\Delta N_l(\%) = (N_{\text{ave}\_l}/N_c) \times 100$ and $\Delta N_{\text{wbr}}(\%) = (N_{\text{ave\_wbr}}/N_{\text{wbr}}) \times 100$. To provide the same test quality for core-based SOCs, a high number of cores (approximately 40%) does not need to be wrapped with WBR cells. Based on functional interconnect topology, there are cases where the maximum number of light-wrapped cores can be half of the total number of cores (see column 3 in Table II). More importantly, the number of WBRs that can be removed varies from hundreds for smaller benchmarks to thousands for the larger ones. Given the fact that each WBR can have an equivalent of 10 to 60 logic gates [25] (depending on the number of flip-flops and the modes used for each cell), we believe that the proposed solution can yield significant savings in DFT area. Because these savings do not come at no expense, the implications on the testing time are discussed next.

C. Experiment 3: Comparison of Test Application Time

To investigate the implications on test application time we compare the results of the modular SOC architecture from Section IV against the case when the light-wrapped cores are tested sequentially using serial EXTEST after the test of all the wrapped cores. When the light-wrapped cores are tested sequentially, it is assumed that the entire parallel TAM bandwidth is allocated to their internal scan chains. It is very important to note, that the use of the optional parallel EXTEST feature of P1500 for producer/consumer loading may improve the scan time if the SOC integrator does not wish to use the proposed producer-CUT-consumer TAM division. However, parallel EXTEST is not applicable to the Test Bus architecture used in this paper, since it requires a TestRail architecture for the wrapped cores [4]. Unwrapping cores in a TestRail architecture and exploiting parallel EXTEST features for loading producers/consumers, is the topic of a completely separate investigation, which is currently undertaken by the authors.

Tables III–VI present test application time results when varying the total TAM width W_{ttl} (note, only results with optimal TAM divisions are reported). $T_{\text{ave\_se}}$, $T_{\max\_\text{se}}$, and $T_{\min\_\text{se}}$ denote the average, maximum, and minimum TAT

TABLE II
NUMBER OF LIGHT-WRAPPED CORES FOR BENCHMARK SOCs

SOC	N_c	$N_{\max\_l}$	$N_{\min\_l}$	N_{ave_l}	ΔN_l(percent)	N_{wbr}	$N_{\max\_\text{wbr}}$	$N_{\min\_\text{wbr}}$	$N_{\text{ave}\_\text{wbr}}$	ΔN_{wbr}(percent)
g1023	14	7	4	5.77	41.21	3587	2367	504	1479.11	41.24
p34392	19	10	7	8.29	43.63	1884	1436	336	804.22	42.69
p93791	32	16	12	13.51	42.22	7697	4235	1983	2871.82	37.31
t512505	31	16	10	12.62	40.71	8503	5035	1798	3287.67	38.66

TABLE III
TEST APPLICATION TIME COMPARISON FOR g1023

	SOC g1023								
	[12]	Serial EXTEST for Light-Wrapped Cores				Proposed Architecture for Light-Wrapped Cores			
W_{ttl}	T(cc)	$T_{\max\_\text{se}}$(cc)	$T_{\min\_\text{se}}$(cc)	T_{ave_se}(cc)	ΔT_{se}(percent)	$T_{\max\_p}$(cc)	$T_{\min\_p}$(cc)	T_{ave_p}(cc)	ΔT_p(percent)
8	66 423	3 164 423	270 004	1 089 174	+1539.75	362 484	87 109	188 585	+183.92
16	35 156	3 149 277	256 732	1 072 909	+2951.85	180 895	42 895	91 760	+161.01
24	24 018	3 141 727	253 263	1 066 494	+4340.39	120 686	28 234	63 304	+163.57
32	19 620	3 141 727	249 221	1 064 404	+5325.10	93 622	21 665	49 676	+153.19
40	14 794	3 141 727	246 090	1 063 731	+7090.29	75 094	19 567	42 305	+185.96
48	14 794	3 141 727	246 090	1 063 597	+7089.38	64 360	19 054	37 667	+154.61
56	14 794	3 141 727	246 090	1 063 479	+7088.58	58 141	19 054	34 589	+133.80
64	14 794	3 141 727	246 090	1 063 299	+7087.37	52 435	18 265	31 997	+116.28

TABLE IV
TEST APPLICATION TIME COMPARISON FOR p34392

	SOC p34392								
	[12]	Serial EXTEST for Light-Wrapped Cores				Proposed Architecture for Light-Wrapped Cores			
W_{ttl}	T(cc)	$T_{\max\_\text{se}}$ (cc)	$T_{\min\_\text{se}}$ (cc)	T_{ave_se} (cc)	ΔT_{se} (percent)	$T_{\max\_\text{p}}$ (cc)	$T_{\min\_\text{p}}$ (cc)	T_{ave_p} (cc)	ΔT_{p} (percent)
8	2 198 975	21926765	3 186 491	11 207 946	+409.69	6 174 198	2 807 634	4 019 375	+82.78
16	1 075 242	21095812	2 369 160	10 339 305	+861.58	3 148 405	1 396 602	2 049 255	+90.59
24	838 643	20859213	2 183 218	10 248 657	+1122.05	2 163 488	998 371	1 477 729	+76.20
32	544 579	20565149	2 133 872	10 114 582	+1757.32	1 682 451	838 643	1 208 688	+121.95
40	544 579	20565149	2 133 872	10 110 141	+1756.51	1 495 912	601 822	1 064 984	+95.56
48	544 579	20565149	2 133 872	10 110 141	+1756.51	1 393 674	563 150	979 985	+79.95
56	544 579	20565149	2 133 872	10 110 141	+1756.51	1 323 567	544 579	923 876	+69.95
64	544 579	20565149	2 133 872	10 110 141	+1756.51	1 286 667	544 579	895 010	+64.35

TABLE V
TEST APPLICATION TIME COMPARISON FOR p93791

	SOC p93791								
	[12]	Serial EXTEST for Light-Wrapped Cores				Proposed Architecture for Light-Wrapped Cores			
W_{ttl}	T (cc)	$T_{\max\ \_\ \text{se}}$ (cc)	$T_{\min\ \_\ \text{se}}$ (cc)	$T_{\text{ave}\ \_\ \text{se}}$ (cc)	ΔT_{se} (percent)	$T_{\max\ \_\ p}$ (cc)	$T_{\min\ \_\ p}$ (cc)	$T_{\text{ave}\ \_\ p}$ (cc)	ΔT_p (percent)
8	3 642 176	20 864 571	5 302 688	10 878 060	+198.67	7 131 063	4 566 696	5 192 296	+42.56
16	1 901 700	19 538 337	3 858 733	9 513 093	+400.24	3 577 099	2 259 929	2 598 762	+36.65
24	1 233 570	19 052 632	3 306 718	9 059 614	+634.42	2 409 185	1 554 572	1 827 146	+48.12
32	1 052 919	18 943 120	3 132 505	8 936 536	+748.74	1 826 561	1 159 523	1 313 552	+24.75
40	869 430	18 849 275	3 000 622	8 801 580	+912.34	1 426 151	963 158	1 132 204	+30.22
48	640 190	18 644 076	2 842 435	8 658 272	+1252.45	1 210 150	787 964	973 060	+52.00
56	598 231	18 642 092	2 802 460	8 636 190	+1343.62	1 145 604	601 717	795 703	+33.01
64	544 052	18 587 913	2 745 080	8 599 643	+1480.67	1 006 763	583 079	680 408	+25.06

TABLE VI
TEST APPLICATION TIME COMPARISON FOR t512505

	SOC t512505								
	[12]	Serial EXTEST for Light-Wrapped Cores				Proposed Architecture for Light-Wrapped Cores			
W_{ttl}	T (cc)	$T_{\mathrm{max\_se}}$(cc)	$T_{\mathrm{min\_se}}$(cc)	$T_{\mathrm{ave\_se}}$(cc)	ΔT_{se} (percent)	T_{max_p} (cc)	T_{min_p} (cc)	T_{ave_p} (cc)	ΔT_p (percent)
8	23550880	28194691	24501542	26024702	+10.50	30181825	29513530	29758333	+26.36
16	11451554	17259395	13291295	14906314	+30.17	13786212	12024617	12836625	+12.10
24	10530995	17256024	13263325	14891063	+41.40	12697603	10453470	11157350	+5.95
32	6740743	13703678	8066245	9816884	+45.64	8528143	6277675	7005682	+3.93
40	5228420	13703678	7960790	9801760	+87.47	7523723	5228420	5961920	+14.03
48	5228420	13703678	7960790	9801760	+87.47	7523723	5228420	5958941	+13.97
56	5228420	13703678	7960790	9801760	+87.47	7523723	5228420	5958844	+13.97
64	5228420	13703678	7960790	9801760	+87.47	7523723	5228420	5958789	+13.97

for the 100 random circuits when serial EXTEST is used to test the light-wrapped cores. T_{ave_p}, $T_{\max\_p}$, and $T_{\min\_p}$ denote the average, maximum, and minimum TAT for the 100 random circuits when the proposed producer–CUT–consumer architecture is used. The percentage changes are calculated using the formula $\Delta T_{\mathrm{se}}(\%) = ((T_{\mathrm{ave\_se}} - T)/T) \times 100$ and $\Delta T_p(\%) = ((T_{\mathrm{ave}\_p} - T)/T) \times 100$, where T is the test application time result obtained using the algorithm described in [12]. It should be noted that since we did not manually select the d and p parameters, T is slightly different when compared to the result reported in [12]. As seen in all the tables, in almost all the cases, ΔT_{se} is much higher than ΔT_p, especially when the total TAM width W_{ttl} is large. This shows the effectiveness of the proposed test architecture. We can observe that $T_{\mathrm{ave\_se}}$ does not change a lot with the variation of W_{ttl} when serial EXTEST is used for testing light-wrapped cores. This is because the single-bit loading/unloading time for producers/consumers dominates the overall TAT of the SOC and the increase of W_{ttl} does not help in shortening it. While for the proposed producer–CUT–consumer architecture, when W_{ttl} is increased, the algorithm will distribute more TAM lines to the bottleneck TAM group and leads to decreased TAT. One exception in the experiments is when $W_{\mathrm{ttl}} = 8$ for SOC t512505, where the serial EXTEST gives a better result. This is because the number of internal memory elements is much larger than the number of cores' I/Os in this SOC. When W_{ttl} is small, test data transportation into the CUT is the bottleneck. Since the proposed architecture requires at least one TAM line for G_{prod} and one TAM line for G_{cons}, only six TAM lines are left in G_{CUT} to transfer test data to/from the cores' internal memory elements, while all eight TAM lines can be used for the same duty when serial EXTEST is employed.

In can be seen in Tables III–VI that the average increase in TAT over [12] can vary from about 4% to 186% when the proposed architecture is used. For g1023, the penalty is higher than for the other SOCs. This is because, in addition to the reasons analyzed earlier in Experiment 1, the number of in ternal scanned flip flops in g1023 is comparable to the number of the producers'/consumers' outputs/inputs. Hence, a large amount of time is necessary to load/unload test stimuli/responses, which imposes a high number of TAM lines assigned to producer/consumer TAMs. This leads to less TAM lines for CUTs to transport test data to/from the internal scan chains of all the cores. It can also be observed that the difference between the maximum and minimum TAT for different functional interconnect topologies may be very high, which is due to the unbalanced sizes of the cores inside the SOCs. For example, there are three large cores in p34392 (Core_2, Core_{10}, and Core_{18}). When these large cores are light wrapped and the functional interconnect topology causes plenty of test conflicts between them, then the TAT will increase significantly. However, this penalty in TAT can be greatly improved simply by wrapping the large cores (which are involved in many test conflicts) with P1500-compliant wrappers. For SOC p93791, the sizes of the cores are medium and hence no core dominates the whole SOC TAT. As a result, although test conflicts exist between cores, the idle time is not too large (in average the increase in TAT is about 37%). The TAT overhead for SOC t512505 is the smallest (in average about 12%) in the four benchmark SOCs. This is because one large core (Core_{31}) dominates the TAT of the entire SOC and the additional time used to test the other incompatible cores is insignificant.

In summary, removing WBRs to save area will obviously increase the testing time. However, in this section, we have demonstrated with experimental data that, when employing the proposed approach, the increase in testing time can be held and is significantly lower than using serial EXTEST for controlling/observing the inputs/outputs of light-wrapped cores.

VI. CONCLUSION

Unlike in boundary scan-based testing, where chips are manufactured before the board is assembled, in core-based system-on-a-chip (SOC) testing, the system integrator has the option of removing wrapper cells without sacrificing controllability and observability. To exploit this option, this paper has described a modular SOC testing methodology based on light-wrapped cores. The proposed approach is scalable and can be equally applied to both manufacturing test and diagnosis since it exploits only the functional interconnect topology and does not rely on the test data at hand. We have proposed a test access mechanism (TAM) design algorithm based on a division in three separate groups that facilitate concurrent testing of both P1500-wrapped and light-wrapped cores. This will limit the increase in testing time, caused by sharing wrapper cells between cores and, as long as the test schedule will match the capacity of the tester buffers, the penalty in the amount of time the chip spends on the tester will be insignificant. This makes the proposed solution particularly attractive, since it is capable to decrease the design for test (DFT) area requirements in complex SOCs and to reduce the propagation delays between cores, which may improve the SOC's performance.

REFERENCES

[1] J. Aerts and E. J. Marinissen, "Scan chain design for test time reduction in core-based ICs," in *Proc. IEEE Int. Test Conf. (ITC)*, Washington, DC, Oct. 1998, pp. 448–457.

[2] L. Chen and S. Dey, "DEFUSE: A deterministic functional self-test methodology for processors," in *Proc. IEEE VLSI Test Symp. (VTS)*, Montreal, QC, Canada, 2000, pp. 255–262.

[3] I. Ghosh, S. Dey, and N. K. Jha, "A fast and low-cost testing technique for core-based system-chips," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 19, no. 8, p. 863, Aug. 2000.

[4] S. K. Goel and E. J. Marinissen, "Effective and efficient test architecture design for SOCs," in *Proc. IEEE Int. Test Conf. (ITC)*, Baltimore, MD, Oct. 2002, pp. 529–538.

[5] ——, "Control-aware test architecture design for modular SOC testing," in *Proc. IEEE European Test Workshop (ETW)*, Maastricht, The Netherlands, May 2003, pp. 57–62.

[6] ——, "Layout-driven SOC test architecture design for test time and wire length minimization," in *Proc. Design, Automation, and Test Europe (DATE)*, Munich, Germany, Mar. 2003, pp. 738–743.

[7] R. K. Gupta and Y. Zorian, "Introducing core-based system design," *IEEE Des. Test Comput.*, vol. 14, no. 4, pp. 15–25, Dec. 1997.

[8] 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," in *Proc. IEEE Asian Test Symp. (ATS)*, Kyoto, Japan, Nov. 2001, pp. 265–270.

[9] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Co-optimization of test wrapper and test access architecture for embedded cores," *J. Electron. Test., Theory Appl.*, vol. 18, no. 2, pp. 213–230, Apr. 2002.

[10] ——, "Efficient wrapper/TAM co-optimization for large SOCs," in *Proc. Design, Automation, and Test Europe (DATE)*, Paris, France, Mar. 2002, pp. 491–498.

[11] ——, "Integrated wrapper/TAM co-optimization, constraint-driven test scheduling, and tester data volume reduction for SOCs," in *Proc. ACM/IEEE Design Automation Conf. (DAC)*, New Orleans, LA, Jun. 2002, pp. 685–690.

[12] ——, "On using rectangle packing for SOC Wrapper/TAM co-optimization," in *Proc. IEEE VLSI Test Symp. (VTS)*, Monterey, CA, Apr. 2002, pp. 253–258.

[13] 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," in *Proc. IEEE Int. Test Conf. (ITC)*, Baltimore, MD, Oct. 2002, pp. 1159–1168.

[14] S. Koranne, "Formulating SoC test scheduling as a network transportation problem," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 21, no. 12, pp. 1517–1525, Dec. 2002.

[15] W.-C. Lai and K.-T. Cheng, "Instruction-level DFT for testing processor and IP cores in system-on-a-chip," in *Proc. ACM/IEEE Design Automation Conf. (DAC)*, Las Vegas, NV, 2001, pp. 59–64.

[16] 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," in *Proc. IEEE Int. Test Conf. (ITC)*, Washington, DC, Oct. 1998, pp. 284–293.

[17] E. J. Marinissen, S. K. Goel, and M. Lousberg, "Wrapper design for embedded core test," in *Proc. IEEE Int. Test Conf. (ITC)*, Atlantic City, NJ, Oct. 2000, pp. 911–920.

[18] E. J. Marinissen, V. Iyengar, and K. Chakrabarty, *ITC'02 SOC Test Benchmarks Web Site*. [Online]. Available: http://www.extra.research.philips.com/itc02socbenchm/

[19] ——, "A set of benchmarks for modular testing of SOCs," in *Proc. IEEE Int. Test Conf. (ITC)*, Baltimore, MD, Oct. 2002, pp. 519–528.

[20] E. J. Marinissen, R. Kapur, M. Lousberg, T. Mclaurin, M. Ricchetti, and Y. Zorian, "On IEEE P1500's standard for embedded core test," *J. Electron. Test., Theory Appl.*, vol. 18, no. 4/5, pp. 365–383, Aug. 2002.

[21] M. Nourani and C. Papachristou, "Structural fault testing of embedded cores using pipelining," *J. Electron. Test., Theory Appl.*, vol. 15, no. 1, p. 129, 1999.

[22] S. Ravi, G. Lakshminarayana, and N. K. Jha, "Testing of core-based systems-on-a-chip," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 3, pp. 426–439, Mar. 2001.

[23] N. Touba and B. Pouya, "Using partial isolation rings to test core-based designs," *IEEE Des. Test Comput.*, vol. 14, no. 4, pp. 52–59, Dec. 1997.

[24] P. Varma and S. Bhatia, "A structured test re-use methodology for core-based system chips," in *Proc. IEEE Int. Test Conf. (ITC)*, Washington, DC, Oct. 1998, pp. 294–302.

[25] C.-W. Wu, *VLSI Testing and Design for Testability Course—Lecture on Core-Based SOC Testing*. [Online]. Available: http://larc.ee.nthu.edu.tw/~cww/n/625/6251/13SOC0211.pdf

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

[27] Q. Xu and N. Nicolici, "Delay fault testing of core-based systems-on-a-chip," in *Proc. Design, Automation, and Test in Europe (DATE)*, Munich, Germany, Mar. 2003, pp. 744–749.

[28] T. Yoneda and H. Fujiwara, "Design for consecutive testability of system-on-a-chip with built-in self testable cores," *J. Electron. Test., Theory Appl.*, vol. 18, no. 4/5, pp. 487–501, Aug. 2002.

[29] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing embedded-core-based system chips," *IEEE Computer*, vol. 32, no. 6, pp. 52–60, Jun. 1999.

[30] W. Zou, S. M. Reddy, I. Pomeranz, and Y. Huang, "SOC test scheduling using simulated annealing," in *Proc. IEEE VLSI Test Symp. (VTS)*, Napa Valley, CA, Apr. 2003, pp. 325–330.

Qiang Xu (S'03) received the B.E. and M.E. degrees in telecommunication engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 1997 and 2000, respectively, and the Ph.D. degree in electrical and computer engineering from McMaster University, Hamilton, ON, Canada, in 2005.

He is currently an Assistant Professor of Computer Science and Engineering at the Chinese University of Hong Kong, Hong Kong. His research interests lie in the broad area of computer-aided design with special emphasis on test and debugging of system-on-a-chip integrated circuits.

Dr. Xu received the Best Paper Award for the 2004 IEEE/Association for Computing Machinery (ACM) Design, Automation and Test in Europe (DATE) Conference and Exhibition.

Nicola Nicolici (S'00–M'00) received the Dipl.Ing. degree in computer engineering from the University of Timisoara, Timisoara, Romania, in 1997, and the Ph.D. degree in electronics and computer science from the University of Southampton, Southampton, U.K., in 2000.

He is an Assistant Professor of Computer Engineering at McMaster University, Hamilton, ON, Canada. His research interests are in the area of computer-aided design and test. He has authored a number of papers in this area.

Dr. Nicolici is a Member of the Association for Computing Machinery Special Interest Group on Design Automation (ACM SIGDA) and the IEEE Computer and IEEE Circuits and Systems Societies and serves on the Editorial Board of IEE Proceedings—Computers and Digital Techniques. He received the IEEE Test Technology Technical Council (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 (2004).