Modular and Rapid Testing of SOCs With Unwrapped Logic Blocks

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

***Abstract*—Extensive research has been carried out for test planning of core-based system-on-a-chip devices. Most of the prior work assumes that all of the embedded cores are wrapped for test purpose. However, some designs may contain user-defined logic or cores that cannot be wrapped due to area constraints or timing violations. This paper discusses how these unwrapped logic blocks can be tested rapidly by adapting the TestRail architecture, which uses only the test control mechanism and the test instructions available through the IEEE 1500 standard for embedded core test. A new test scheduling algorithm, which facilitates a concurrent test of both unwrapped logic blocks and IEEE 1500-wrapped cores, is proposed, and experiments show that it outperforms a previous approach when the available number of tester channels and/or the number of unwrapped logic blocks are small.**

***Index Terms*—Light-wrapped cores, system-on-a-chip (SOC) testing, test scheduling.**

I. Introduction

GIVEN THE increase in the development time of complex integrated circuits, a new category of devices, called system-on-a-chip (SOC) devices, have emerged. They consist of predesigned and preverified blocks called intellectual property (IP) cores. Although core reuse increases the productivity, due to the growing transistor-to-pin ratio and the increasing number of embedded cores that are not directly accessible from the input/output (I/O) ports of the SOC, the manufacturing test is posing a design implementation problem. Various solutions for exploiting the SOC's architecture-specific information and using functional interconnect as test access mechanisms (TAMs), either at the core or system level, have been proposed [4], [5], [8], [22], [29], [31], [42]. 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, or interoperability and may become the computational bottleneck in the test automation flow. This problem is overcome by the modular test strategies [44], 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. The IEEE 1500 Standard for Embedded Core Test (SECT) intends to facilitate reuse for SOC testing [12], [28]. Its architecture consists of test control lines, TAM, and core wrappers. The test control lines set the operational mode of each core and the TAM is used to transfer data to/from the core-under-test (CUT) from/to primary inputs (PIs)/primary outputs (POs). The core wrapper comprises an instruction register, control logic, and wrapper boundary register (WBR) cells, which facilitate an isolation mechanism in the test mode and provide full controllability and observability to the core terminals.

SOCs contain user-defined logic (UDL), which is necessary for custom core integration and product differentiation. Although the system integrator has the full knowledge of the UDL's structure and functionality, the manufacturing test of UDL suffers from the same test access problem as the embedded core test. On the one hand, a small part of the UDL implements dedicated logic functions, which can be reused for future designs. Hence, this type of UDL can be integrated and tested as an in-house developed embedded core. On the other hand, UDL comprises also glue logic used for interfacing different cores or implementing SOC-specific functions. This type of UDL does not need to be reused, and it is difficult to be wrapped due to its high number of I/O terminals, which will ultimately incur a large area overhead associated with all of the WBR cells. In addition, there are also reusable cores that, if placed on the critical paths of the design, the multiplexers in WBR cells will lower the maximum operating frequency, thus having a direct impact on the system's performance.

The UDL or IP cores that cannot be wrapped due to area constraints or timing violations are referred to as *unwrapped logic blocks* [36]. Since the gate count of these blocks can be large, the question is how can they be tested effectively? One approach is to treat them as interconnect circuitry in between wrapped cores and test them as nonscanned sequential logic, as discussed in [23] and [25]. If there is a large number of flip-flops and/or latches in the unwrapped logic blocks, this approach may present several problems. First, when compared to the scanned version of the unwrapped logic blocks, the fault coverage may significantly decrease despite the increased computational time required for sequential automatic test pattern generation (ATPG). This is unacceptable, according to the analysis given in [19], where it was shown that 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. Second, due to sequential ATPG, the test pattern count will grow, which may also lead to an increase in the overall

Manuscript received June 6, 2005; revised August 25, 2005. This work was supported in part by Micronet and Gennum Corporation. Preliminary versions of this paper were published in the informal proceedings of the IEEE European Test Symposium (ETS), May 2005, and in the formal proceedings of the IEEE International Test Conference (ITC), November 2005.

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 Computer-Aided Design and Test Group, Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON L8S 4K1, Canada (e-mail: nicola@ece.mcmaster.ca).

Digital Object Identifier 10.1109/TVLSI.2005.859585

1063-8210/$20.00 © 2005 IEEE

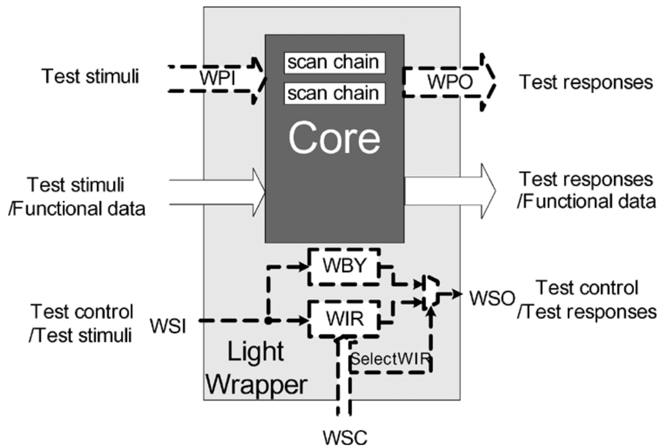

Fig. 1. Light-wrapped core without WBR cells [39].

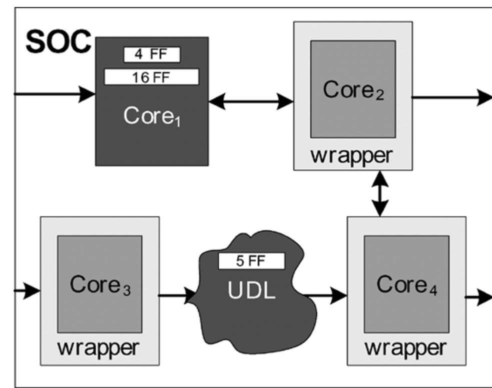

Fig. 2. SOC with light-wrapped cores.

testing time because the test stimuli/responses are not directly controlled/observed (they have to be shifted in/out through other cores' WBR cells). The research presented in [30] and [34] proposed to justify the unwrapped logic block's test vectors through its surrounding UDL, without affecting the fault coverage. Although this solution is effective in reducing the DFT overhead, its main limitation lies in the fact that it reduces the reusability of the core test sets, since new test sets are required whenever the unwrapped logic block is used in a different SOC environment.

The aim of this paper is to present a new effective and efficient test strategy that facilitates concurrent test of unwrapped logic blocks and wrapped cores. The remainder of this paper is organized as follows. Section II reviews prior work in this domain and summarizes the main contributions of this paper. In Section III, we present our proposed TAM design and its corresponding test scheduling algorithm. Section IV contains the experimental results for benchmark SOCs from the ITC'02 benchmark set [27]. Finally, Section V concludes this paper.

II. Preliminaries and Motivation

To the best of our knowledge, [39] provides the only modular approach, reported in the public domain, for *concurrent* test of wrapped cores and unwrapped logic blocks. Despite its effectiveness for increasing fault coverage for unwrapped logic blocks, [39] has been designed *only* for the Test Bus architecture [35] and, consequently, it requires extra test instructions and it increases test control complexity. Following a short review of light-wrapped cores, the main modular test access architectures are summarized and the motivation of this work is given.

A. Light-Wrapped Cores

Light-wrapped cores, as depicted in Fig. 1, are IP cores without input and output WBR cells. If the core does not have other test modes except INTEST and EXTEST mode, then it does not need a wrapper at all. However, if the core has other test modes (e.g., built-in self-test mode to test internal memories inside the core), then a light wrapper that includes a WIR and the WSC port to control the operational mode of the core is needed. There are two important properties to be satisfied: 1) the light-wrapped cores must be surrounded by IEEE 1500-wrapped cores (which could be many) and, by reusing the functional interconnect for transferring test data from and to its producers and consumers,[1] full controllability and observability can still be provided and 2) for light-wrapped cores with internal scan chains, these scan chains need to be driven from dedicated test access lines and a bypass mechanism is provided to shorten the test access between different light-wrapped cores or between light-wrapped cores and wrapped cores. To facilitate modular SOC testing, all of the unwrapped logic blocks (including unwrapped UDL) can be treated as light-wrapped cores.

A hypothetical SOC with light-wrapped cores is shown in Fig. 2. For testing the internal logic of Core_1 and the UDL, the controllability of its input terminals can be provided through its producers' output WBR cells and/or the primary inputs of the SOC, while the observability of the output terminals is provided through its consumers' input WBR cells and/or the primary outputs of the SOC. Besides, because the test stimuli and test responses are applied and captured through functional paths, all of the interconnects are tested implicitly, and hence there is no need to perform ExTest for the interconnects driving or driven by Core_1 and UDL. The integration of light-wrapped cores into a modular test infrastructure available for IEEE 1500-wrapped cores is key to lowering the overall test application time for the entire SOC.

One of the design aims in [39] is to maximize the number of light-wrapped cores. This can lead to very long test application time, which, in some situations, may not be the preferred option for the system integrator. One may argue that, from the test reuse standpoint, as long as the existence of the wrappers does not violate the design constraints, it is better to employ them, regardless of the overhead that they may incur. Hence, a more practical situation is that the system integrator analyzes the design requirements and SOC constraints and determines the wrapper type, for each embedded IP core or user-defined block, on a case-by-case basis. It is the above-mentioned situation that motivates the presented research.

[1]For a given Core_i, the producers are the cores which feed its primary inputs and the consumers are the cores which capture its primary outputs in the normal (functional) mode [39].

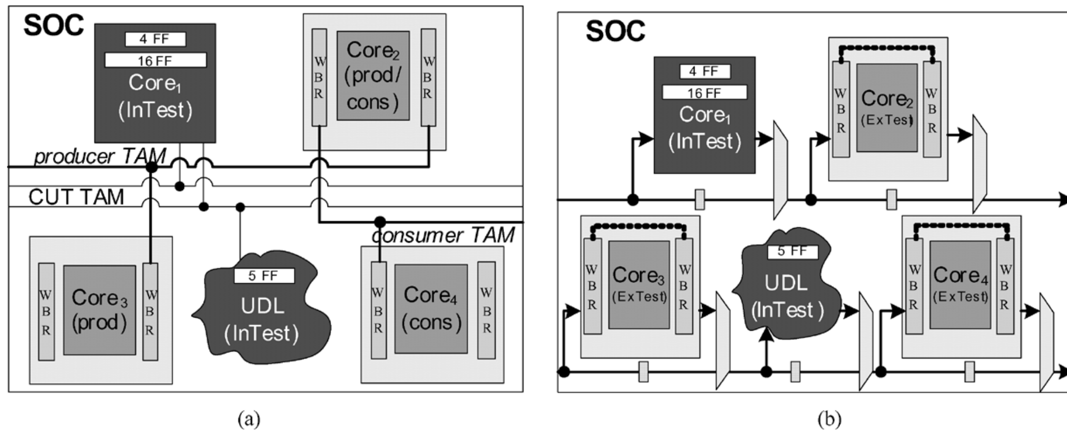

Fig. 3. Test architectures for SOCs containing light-wrapped cores—a simple example. (a) Producer/CUT/consumer model for Test Bus. (b) TestRail with NO added control.

B. Related Work

Test architectures for modular SOC testing have been subject to extensive research in recent years [1], [26], [35]. Three basic types of test architectures are [1]: 1) the *Multiplexing* architecture; 2) the *Daisychain* architecture; and 3) the *Distribution* architecture. In the Multiplexing and the Daisychain architectures, all cores get full access to all TAMs, while in the Distribution architecture each core gets assigned its private TAM lines. Two of the most well-known combinations of the above architectures, which are necessary to support more flexible test schedules, are Test Bus and TestRail architectures. The Test Bus architecture [35] can be seen as a combination of the Multiplexing and Distribution architectures. Multiple multiplexed test buses on an SOC operate independently (as in the Distribution architecture) and, hence, allow for concurrent testing of wrapped cores. However, its main limitations are that cores connected to the same Test Bus can only be tested sequentially. and there is no implicit support for parallel test data transfer for ExTest. The TestRail architecture [26] is a combination of the Daisychain and Distribution architectures. Multiple TestRails on an SOC also operate independently; however, unlike for the Test Bus architecture, cores on each individual TestRail can be tested both sequentially and concurrently. This also facilitates external testing using parallel TAM lines.

The bus-based TAM architecture (i.e., Test Bus or TestRail architecture) can be further categorized into two types [15]:

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

Numerous TAM design and test scheduling algorithms [41] have been researched to reduce SOC testing time, assuming that all cores and UDL are IEEE 1500-wrapped. For the fixed-width architecture, Iyengar *et al.* [13] first formulated the integrated wrapper/TAM co-optimization problem and solved it using the integer linear programming (ILP) technique. To decrease the CPU running time, the same authors [14] proposed to combine efficient heuristics and ILP methods. In [32], Sehgal *et al.* presented an optimization method based on Lagrange multipliers and achieved better results. Koranne [20] formulated the test scheduling problem as a network transportation problem and proposed a two-approximation algorithm to solve it. Goel and Marinissen [6] tackled the same problem with a new efficient heuristic algorithm TR—*Architect*, which works for both cores having fixed-length and cores having flexible-length scan chains. They also presented the lower bounds for SOC testing time in this work. For the flexible-width architecture, the optimization problem was mapped to the well-known two-dimensional (2-D) bin packing problem in [9], [10], [17], and [45] and solved with different heuristics. A p-admissible representation of SOC test schedules based on the use of k-tuples is presented in [21], and a greedy random search algorithm was used to find an optimal test schedule. In addition to the above, there are also many other methods (e.g., [3], [11], [16], [24]–[26], [33], [40], and [43]) proposed in the literature.

Although the above techniques mainly consider optimizing the test architecture of SOCs with all logic blocks IEEE 1500-wrapped, they can be adapted to test SOCs containing light-wrapped cores as well, in which case test access resources must be shared and test scheduling algorithms for wrapped cores need to be adapted. If the unwrapped logic blocks are in a separate test session, then either serial ExTest, for the Test Bus, or parallel ExTest, for the TestRail, can be used. In this case, the control mechanism of the IEEE SECT can be reused; however, if the size of the unwrapped logic blocks increases, the testing time may become prohibitively large.

In [39], a *producer/CUT/consumer* model based on the Test Bus architecture was introduced, which is able to test IEEE 1500-wrapped and light-wrapped cores concurrently. As shown in Fig. 3(a), the authors proposed to divide the total TAM into CUT, producer, and consumer TAM groups. These TAM groups are then used to scan in/out the internal scan chains of the CUT, the POs of the producers, and the PIs of the consumers, respectively. One of the main limitations of this solution is its control complexity, because three separate test access architectures are required and new test modes need to be introduced to the cores to act as producers and/or consumers. In addition, the test architecture described in [39] is less efficient when the total number

of TAM lines is small, because, in such a case, producer and consumer TAMs normally sit idle for a long time and hence these test resources are wasted. When multisite testing is utilized to reduce the SOC test cost [7], [18], [37], the number of TAM lines is usually small, which necessitates a better test strategy for this reduced-pin-count testing scenario [2], [38]. Furthermore, the test scheduling algorithm in [39] is more effective when the number of unwrapped logic blocks is large. In practice, however, the number of unwrapped logic blocks is usually **not** large since the system integrator tends to wrap as many cores as possible to reduce the SOC testing time, as long as the DFT logic does not affect the timing or the area constraints.

By adapting the TestRail architecture and using a new test scheduling algorithm, as shown in the next section, the approach proposed in this paper is capable of combining the benefits of achieving an effective schedule when the number of unwrapped logic blocks and/or the number of TAM lines is small and the simple control mechanism available through the mandatory test instructions (e.g., InTest, ExTest, and Bypass) of the IEEE SECT.

III. TAM Design and Test Scheduling

Since the TestRail architecture supports loading cores on individual TestRails both sequentially and concurrently, it automatically supports the access of a light-wrapped core's producers and consumers. Therefore, no dedicated TAM resources are necessary for shifting in/out producers' outputs and consumers' inputs, as it is the case in [39]. For the TestRail architecture, when a light-wrapped core is under test, we simply set its producers and consumers in the ExTest mode and set itself in the InTest mode if it is a scanned core. That is, TestRails are constructed to walk through not only the CUT's I/Os and internal scan chains, but also the I/Os of its producers and consumers, as shown in Fig. 3(b). No new test modes, and hence no additional test commands, need to be introduced in the wrapper instruction set. When using the TestRail architecture, both the inputs and outputs of the producers and consumers are loaded through the TestRails used by the wrapped cores. Therefore, we do not need to differentiate them and, from now on, they are both regarded as the test partners of the light-wrapped cores. Because the light-wrapped cores and their test partners might be placed in different TestRails, separate TestRails can no longer operate independently. This necessitates a new TAM design and test scheduling algorithm, which will be described in this section.

A. Problem Definition and Top-Level Algorithm

The problem of minimizing test application time of the TestRail architecture for SOCs with light-wrapped cores, called $P_{\mathrm{lightTR-opt}}$, can be formulated as follows.

Problem $P_{\mathrm{lightTR-opt}}$: Given the test set parameters for each core and UDL (including the number of primary inputs, primary outputs, bidirectional I/Os, number of test patterns, number of scan chains, scan chain lengths for cores with fixed-length scan chains, and number of scan cells for cores/UDL with flexible scan chains), the wrapper property of each core and UDL (light-wrapped or IEEE 1500-wrapped), the total TAM width $W_{\max}$ for the SOC, determine the set of TestRails R, the width $w(r)$ of each TestRail r, the wrapper design for each core c, and a test schedule for the entire SOC such that: 1) every core or UDL is assigned to not more than one TestRail; 2) $\sum_{r\in R} w(r) \leq W_{\max}$; and 3) the overall SOC test application time T_{soc} is minimized.

Algorithm 1 - LightTRDesign

INPUT: C, W_{max}
OUTPUT: R, T_{soc}

```
1. for(i = 0; i < loopCnt; i++) {
2.   BuildCostFunction;
3.   DesignTestRail;
4.   ScheduleLightCores;
5.   RescheduleInRail;
6.   RescheduleBetweenRails;
7.   record the TestRail architecture R with lowest T_soc;
. }
8. return R, T_soc;
```

Fig. 4. Pseudocode for optimizing TestRail architecture with light-wrapped cores.

The total test application time T_{soc} for the TestRail architecture is the maximum of the test application times of all the individual TestRails. The IEEE 1500-wrapped cores connected to a TestRail r are assumed to be tested sequentially, i.e., while a core is tested, all of the other cores connected to the same TestRail are bypassed. The light-wrapped cores can be either tested sequentially or concurrently, depending on which alternative saves testing time. It is important to note that light-wrapped cores that do not have internal scan chains do not need to be assigned to any TestRail because their test stimuli/responses can be fully controlled/observed from their test partners. The test application time of a light-wrapped core cannot be determined until all its test partners (which are IEEE 1500-wrapped cores) have already been assigned to dedicated TestRails. To decrease the complexity of the $P_{\mathrm{lightTR-opt}}$ problem, we first separate the schedules for IEEE 1500-wrapped cores and light-wrapped cores and then merge them at a later stage of the algorithm.

The proposed top-level algorithm *LightTRDesign* is a loop procedure with an upper limit on the exploration time decided by the predefined value (*loopCnt*). It takes the SOC core set C and the total TAM width $W_{\max}$ as inputs, and it outputs the set of TestRails R and the overall test application time T_{soc}. *LightTRDesign* (shown in Fig. 4) has four main steps (lines 3–6). For each of the iterations, the cost function is different which implies that we explore *loopCnt* number of different TestRail Architectures and select the one which leads to the lowest test application time. This is important because the initial test architecture determines the effectiveness of the subsequent optimization procedures. In the results reported in this paper $\mathrm{loopCnt} = 500$, which gives a good balance in between the quality of the results and CPU execution times that are in the seconds range. Step *DesignTestRail* determines a set of TestRails and their widths, based on the cost function generated from *BuildCostFunction*. Step *ScheduleLightCores* schedules the light-wrapped cores in front of IEEE 1500-wrapped cores for the previously determined TestRail architecture. The last two steps (*RescheduleInRail* and *RescheduleBetweenRails*) try to optimize the overall SOC test application time for the SOC by exploiting the idle times within TestRails. In the following,

TABLE I
DATA STRUCTURE

Data structure core schedule

1. *TestRail*; /* The TestRail that the core is on */
2. *begin*; /* Schedule begin time of the core */
3. *end*; /* Schedule end time of the core */
4. *isScheduled*; /* Whether the core has been scheduled */
5. *inCompatibleCores*; /* The cores that cannot be scheduled concurrently */
6. *finishedPatterns*; /* The number of patterns that finished schedule*/
7. *finishedTime*; /* The test application time of the finished patterns*/

we illustratively show the specific features of our overall algorithm. The four main steps (lines 3–6) are described in detail using a hypothetical SOC afterwards.

Test Conflicts: Because testing the internal logic of the light-wrapped cores uses the IEEE 1500-wrapped test partners, two new types of test resource conflicts are introduced.

- *Partner-CUT Conflict*: The light-wrapped CUT and its test partners **cannot** be tested at the same time. This is because both of them need to utilize the WBRs of the test partner to shift in/out test stimuli/responses.
- *Shared-Partner Conflict*: Two light-wrapped cores which connect directly (i.e., on a dedicated nonshared set of lines) to the same test partner **cannot** be tested at the same time. This is because, as in the above case, sharing of the partner's WBRs for test data transfer is prohibited.

Data Structure: The data structure used to store the schedule information for each core is given in Table I.

The *inCompatibleCores* list for every core is initialized in a preprocessing step based on the functional interconnects and wrapper properties. For an IEEE 1500-wrapped core, once it was assigned to a TestRail, its test application time is determined, however, its schedule sequence on the TestRail is not yet decided. We still consider it unscheduled ($\mathrm{isScheduled} = \mathrm{false}$) and therefore its *begin* and *end* times will be updated after the schedule of light-wrapped cores has been resolved. Since the test of light-wrapped cores involves its test partners, a light-wrapped core schedule might affect many other cores scheduled at the same time. Hence, when a light-wrapped core i has completed its schedule, *finishedPatterns* and *finishedTime* at end (i) are updated for all the other concurrently scheduled cores.

Test application time for light-wrapped cores: When a new light-wrapped core is scheduled, all of the cores that are tested at the same time as this light-wrapped core will change their test application time. This is illustrated using the following example.

Example 1: The hypothetical SOC shown in Fig. 2 contains two light-wrapped cores: Core_1 and the UDL. When using TestRail architecture shown in Fig. 3(b), based on the functional interconnect, Core_2 is test partner of Core_1, while Core_3 and Core_4 are test partners of UDL. Suppose the test pattern count for Core_1 and UDL are N_1 and N_{udl}, respectively ($N_1 > N_{\mathrm{udl}}$). The TestRail architecture can be determined by the procedure DesignTestRail and, as shown in Fig. 5(a), Core_1 will be assigned to TestRail 1. The rectangle Core_1 shown in the figure includes the time to load/unload both Core_1 and its test partner Core_2, i.e., the load size for each pattern of Core_1 is $\mathrm{loadsize}_1 = N_{\mathrm{sc1_}W_1} + \lceil N_{\mathrm{io2}}/W_1 \rceil + 1$,[2] where $N_{\mathrm{sc1_}W_1}$ is the maximum scan chain length of Core_1 after scan chain stitching to match TAM width W_1 and N_{io2} is the number of wrapper boundary cells of its test partner Core_2 (the value 1 stands for the bypass cycle on Core_4). Since Core_1 and Core_2 are on the same TestRail 1, the schedule of Core_1 is independent of TestRail 2. If the UDL is assigned on TestRail 1, as shown in Fig. 5(b), since it shares the same TestRail with Core_1 and they are tested concurrently, the load size for each of the overlapped test patterns is $\mathrm{loadsize}_{\mathrm{overlapped}} = \max\{N_{\mathrm{sc1_}W_1} + N_{\mathrm{scud}l_W1} + \lceil N_{\mathrm{io2}}/W_1 \rceil + \lceil N_{\mathrm{io4}}/W_1 \rceil, \lceil N_{\mathrm{io3}}/W_2 \rceil\}$, where $N_{\mathrm{scudl_}W_1}$ stands for the maximum scan chain length of UDL after scan chain stitching to match TAM width W_1. If the UDL is assigned on TestRail 2, as depicted in Fig. 5(c), however, although Core_1 and UDL are not on the same TestRail, their test partners share the same TestRail. Therefore, the load size for each of the overlapped test patterns will be $\mathrm{loadsize}_{\mathrm{overlapped}} = \max\{N_{\mathrm{sc1_}W_1} + \lceil N_{\mathrm{io2}}/W_1 \rceil + \lceil N_{\mathrm{io4}}/W_1 \rceil, N_{\mathrm{scud}l_W_2} + \lceil N_{\mathrm{io3}}/W_2 \rceil\}$. In Fig. 5(b) and (c) the rectangle $\mathrm{UDL}_{\mathrm{tp}}$ stands for the time required to load the test patterns of UDL. In both cases after the UDL has finished its schedule, the load time for each of the remaining $N_1 - N_{\mathrm{udl}}$ test patterns for Core_1 is $\mathrm{loadsize}_1$.

Hypothetical SOC for test scheduling: To better illustrate the major steps of the proposed test scheduling algorithm *LightTRDesign* in the following sections, we provide a hypothetical SOC, called lightSOC, with five IEEE 1500-wrapped cores and three light-wrapped cores (predefined by the system integrator), as shown in Fig. 6.[3]

B. Determine the TestRail Architecture

In procedure **DesignTestRail**, we determine the set of TestRails R and the width of each TestRail $w(r)$ by optimizing TestRail architecture only for IEEE 1500-wrapped cores. Since the testing time of light-wrapped cores is dependent on its test partners' shifting time, we need to consider it in this step in order to get an initial TestRail architecture more suitable for assigning light-wrapped cores in the following steps of the top-level Algorithm 1. Therefore, we use *BuildCostFunction* to try different costs in every iteration of Algorithm 1. Given a TestRail r with IEEE 1500-wrapped cores C and for each core $c_i \in C$ the time it serves as test partners is t_i and its number of wrapper cells is N_w, the cost function is $\mathrm{TtlCost} = T(r) + \alpha \times \sum_i (t_i \times N_w)$, where $T(r)$ is the test application time for the TestRail, and α is a cost weighting scalar that is varied by the system integrator in solution space exploration. In our implementation, α is selected to be incremented by 0.1 in each iteration (e.g., for a $\mathrm{loopCnt} = 500$, α varies from 0 to 49.9), which gives us good results with execution time of seconds.[4]

[2]Wrapper scan chains are built by concatenating core internal scan chains and WBR cells.

[3]PC stands for IEEE 1500-wrapped core, while LC denotes light-wrapped core.

[4]We have also tried with $\alpha = 0.01$ and $\mathrm{loopCnt} = 5000$ in our experiments, and the average improvement in terms of testing time when compared to $\alpha = 0.1$ and $\mathrm{loopCnt} = 500$ is significantly less than one percent.

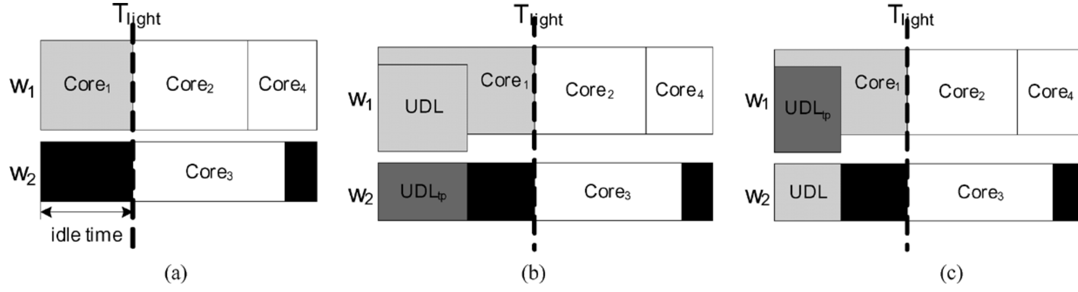

Fig. 5. Schedule of UDLs on different TestRails. (a) UDL to-be-scheduled. (b) UDL scheduled on TestRail 1. (c) UDL scheduled on TestRail 2.

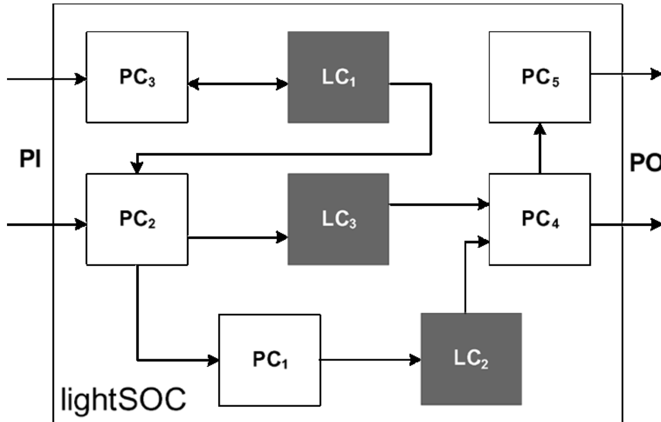

Fig. 6. Example SOC for TAM design and test scheduling.

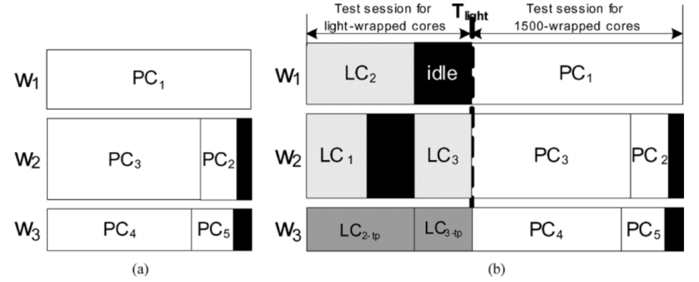

Fig. 7. Test schedule for *lightSOC* after (a) *DesignTestRail* and (b) *ScheduleLightCores*.

TR-Architect [6] is revised to optimize *TtlCost* instead of testing time only in procedure DesignTestRail for the test architecture exploration. Therefore, we briefly discuss this algorithm here. For the sake of self-containment, however, please refer to [6] for more details and terminology. *TR-Architect* has four main steps. The basic idea is to divide the total TAM width over multiple cores based on their test data volume. The algorithm first creates an initial test architecture by assigning value 1 to each core's TAM width. Since the overall test application time of the SOC $T_{\rm soc}$ equals the *bottleneck* TAM with the longest test application time, in the second and the third steps, the algorithm iteratively optimizes $T_{\rm soc}$ through merging TAMs and distributing freed TAM resources. Either two *nonbottleneck* TAMs are merged with less TAM width to release freed TAM resources to the bottleneck TAM, or the bottleneck TAMs is merged with another TAM to decrease $T_{\rm soc}$. In the last step, the algorithm tries to further minimize $T_{\rm soc}$ by placing one of the cores assigned to the bottleneck TAM to another TAM.

It should be noted that, after step *DesignTestRail*, the number of TestRails and the width of each TestRail cannot be changed, only the cores assigned to each TestRail can be modified by the following steps. The variation of *TtlCost* in each iteration, however, leads to different initial test architectures and hence increases the flexibility of our test scheduling algorithm.

For the example *lightSOC* depicted in Fig. 6, after the *DesignTestRail* step, its test schedule is shown in Fig. 7(a). It can be seen that only IEEE 1500-wrapped cores are scheduled on them to determine the initial TestRail architecture, and there are three TestRails in this example.

C. Schedule Light-Wrapped Cores

The procedure *ScheduleLightCores*, as shown in Fig. 8, schedules the light-wrapped cores onto a given TestRail architecture and tries to reduce the overall test application time. It takes the set of TestRails R and the light-wrapped core set $C_{\rm light}$ as inputs, and it outputs the updated TestRail set R' with all the light-wrapped cores scheduled on them and the overall test application time of the light-wrapped cores $T_{\rm light}$. Algorithm 2 lists the pseudocode for this procedure. In this procedure, we schedule all the light-wrapped cores in front of IEEE 1500-wrapped cores in each TestRail, and there is no schedule overlap between any IEEE 1500-wrapped cores and light-wrapped cores so that we do not need to consider the Partner-CUT conflicts. In lines 1 and 2, we initialize R', the unscheduled core set $C_{\rm unScheduled}$ and the currently scheduled core set $C_{\rm scheduling}$. Inside the loop, the light-wrapped cores are scheduled (line 3–23). The procedure first finds a core i compatible with $C_{\rm scheduling}$ (i.e., it does not have any shared-partner conflicts with the cores in $C_{\rm scheduling}$) with maximum test pattern count (line 4). Then if core i is a non-scanned core, it will not be assigned to any TestRail. In this case the procedure only updates the schedule of all the affected cores (line 6–7). If core i is a scanned core, the procedure will search through all the TestRails and try to assign the core to the TestRail r^* which leads to the minimum test application time. The time must account for the sum of all of the affected light-wrapped cores $C_{\rm affected}$ and the maximum test application time $T(R_{\rm affected})$ for all the IEEE 1500-wrapped cores on the affected TestRails $R_{\rm affected}$ (lines 10–17). Taking $T(R_{\rm affected})$ into consideration is very important since it helps us to avoid the assignment of the light-wrapped cores which will affect badly the TestRail that already has a large testing time for the IEEE 1500-wrapped cores. After scheduling core $i, C_{\rm scheduling}$ and $C_{\rm unScheduled}$ are updated (lines 18–19). If a compatible core with $C_{\rm scheduling}$ cannot be found, the schedule of core j in $C_{\rm scheduling}$ with the minimum end time will be finished, the number of finished test patterns and test application time of all the other cores in $C_{\rm scheduling}$ are updated (lines 21–23).

Algorithm 2 - ScheduleLightCores

INPUT: R, C_{light}
OUTPUT: R', T_{light}

1. set $R' = R$;
2. set $C_{unScheduled} = C_{light}$; $C_{scheduling} = \emptyset$;
3. **while**($C_{unScheduled} != \emptyset$) {
4.   **if** a compatible core i can be found with maximum N_p {
5.   **if** (core i is non-scanned core) {
6.   find the cores $C_{affected}$ whose schedule are affected;
7. $compTime(C_{affected}, R')$;
8.   } **else** {
9.   Set $t(r^*)$ maximum value;
10.   **for all** $r \in R'$ {
11. $r_{temp} = r \cup \{i\}$; $R_{temp} = R' \setminus \{r\} \cup \{r_{temp}\}$;
12.   find the TestRails $R_{affected}$ whose schedule are affected;;
13.   find the cores $C_{affected}$ whose schedule are affected;
14. $t(r_{temp}) = compTime(C_{affected}, R_{temp}) + T(R_{affected})$;
15.   **if**($t(r_{temp}) < t(r^*)$) {
16. $r^* = r_{temp}$; $R^* = R_{temp}$;
.   }
.   }
17. $R' = R^*$;
.   }
18. $C_{scheduling} = C_{scheduling} \bigcup \{i\}$;
19. $isScheduled(i) = true$; $C_{unScheduled} = C_{unScheduled} \setminus \{i\}$;
20.   } **else** {
21.   find core j in $C_{scheduling}$ such that $end(j)$ is minimum;
22. $C_{scheduling} = C_{scheduling} \setminus \{j\}$;
23.   update $N_{finished}$, $T_{finished}$ for all cores in $C_{scheduling}$;
24.   }
. }
25. set $T_{light} = max_{i \in C_{light}} end(i)$;
26. **return** R', T_{light};

Fig. 8. Procedure for scheduling light-wrapped cores onto TestRail architecture.

Algorithm 3 - RescheduleInRail

INPUT: R, T_{light}
OUTPUT: R'

1. **for all** $r \in R'$ {
2.   set $upperLimit = 0$;
3.   **while** (**true**) {
4.   **if** an idle range $< idleBegin, idleEnd >$ on r can be found
.   such that $idleBegin \geq upperLimit$ {
5.   **if** ($idleEnd == T_{light}$) {
6.   set T_{idle} maximum value;
7.   } **else** {
8. $T_{idle} = idleEnd - idleBegin$;
.   }
9.   **if** a unscheduled IEEE 1500-wrapped core i in r can be found
.   such that $T_i < T_{idle}$ AND T_i is the maximum {
10. $begin(i) = idleBegin$; $end(i) = begin(i) + T_i$;
11. $upperLimit = end(i)$;
12.   } else {
13. $upperLimit = idleEnd$;
.   }
14.   } **else** {
15.   **break**;}
16.   update the schedule of the remaining IEEE 1500-wrapped cores;
.   }
. }
17. **return** R';

Fig. 9. Procedure for rescheduling IEEE 1500-wrapped cores within TestRail.

The procedure is repeated until all the light-wrapped cores are scheduled. In line 25, the overall test application time of light-wrapped cores T_{light} is computed, which also gives the begin time for IEEE 1500-wrapped cores in every TestRail.

For the example *lightSOC* depicted in Fig. 6, after the *ScheduleLightCores* step, its test schedule is shown in Fig. 7(b). It can be seen the overall test schedule is divided into separate test sessions for light-wrapped cores and IEEE 1500-wrapped cores, and hence incurs a relatively large idle time. It can be also observed that since LC_2 and LC_3 share the same test partner PC_4, and LC_1 and LC_3 share the same test partner PC_2, they are not tested concurrently.

D. ReSchedule Wrapped Cores Within TestRail

The procedure **RescheduleInRail**, shown in Fig. 9, tries to move the IEEE 1500-wrapped cores to the idle time created by scheduling light-wrapped cores, in order to reduce the overall test application time of the longest TestRail. Since the core set on every TestRail will not change, this rescheduling will not affect the core schedule on other TestRails. For each TestRail r, the procedure searches through all the idle ranges one by one (controlled by *upperLimit*), to reschedule the IEEE 1500-wrapped cores. First the idle time is computed (lines 4–8). If the idle range ends at T_{light}, then all the remaining IEEE 1500-wrapped cores can move forward to the beginning of this idle range. Hence T_{idle} is set as maximum value (lines 5–6). Then an unscheduled core i with maximum test application time which can fit in the idle range is found and scheduled (lines 9–11). If such a core cannot be found, $upperLimit$ will be updated to the end time of this idle range so that the procedure will try the next idle range (line 13). Once all the idle ranges are searched, the procedure will update the schedule of the remaining IEEE 1500-wrapped cores (line 16). The procedure for rescheduling IEEE 1500-wrapped cores in between different TestRails is shown in Fig. 10, and it will be discussed in the following subsection.

As illustrated in Fig. 11(a), after the **RescheduleInRail** step, for the example lightSOC depicted in Fig. 6, the IEEE 1500-wrapped cores PC_1 on TestRail1 and PC_2 on TestRail2 are rescheduled. As a consequence, the TestRail3 becomes the new bottleneck TAM, which shortens the overall test application time of the SOC.

E. ReSchedule Wrapped Cores in Between Different TestRails

The procedure **RescheduleBetweenRails** attempts to reduce the test application time of a given TestRail architecture by moving the IEEE 1500-wrapped cores from the bottleneck TestRail to another TestRail, provided that it reduces the overall test application time. If a IEEE 1500-wrapped core serves as

Algorithm 4 - RescheduleBetweenRails

INPUT: R
OUTPUT: R', T_{soc}

1. $isImproved = true$;
2. **while** (isImproved) {
3. find r_{max} for which $T(r_{max}) = max_{r \in R}\ T(r)$;
4. find cores C on r_{max} which do not serve as test partners;
5. **if** $(C == \emptyset)$ {$isImproved = false$; **break**;}
6. find core i^* in C for which $T_{i^*} = min_{i \in C} T(i)$;
7. Set $isFound = true$;
8. **for all** $r \in R \setminus \{r_{max}\}$ {
9. **for all** idle ranges $< idleBegin, idleEnd >$ on r {
10. $T_{idle} = idleEnd - idleBegin$;
11. **if** $(TestTime(i^*, r) \leq T_{idle})$ {
12. schedule core i^* to the idle range;
13. $r_{max} = r_{max} \setminus \{i^*\}$; $r = r \bigcup \{i^*\}$;
14. $isFound = true$;
15. **break**;
. }
. }
16. **if** (isFound) **break**;
. }
17. if (!isFound) {isImproved=false};
. }
18. $T_{soc} = max_{r \in R'} T(r)$;
19. **return** R', T_{soc};

Fig. 10. Procedure for rescheduling IEEE 1500-wrapped cores in between different TestRails.

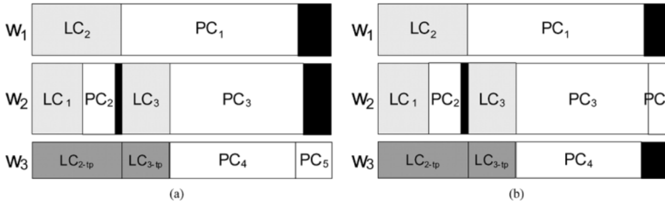

Fig. 11. Test Schedule for *lightSOC* after (a) *RescheduleInRail* and (b) *RescheduleBetweenRails*.

a test partner for light-wrapped cores, placing it to a different TestRail will affect core schedule on other TestRails. As a result, we only consider moving those cores which do not serve as test partners for any light-wrapped cores in this procedure. The procedure is iterative. In each iteration, it first identifies the bottleneck TestRail $r_{\max}$ (line 3). Line 4 finds all the IEEE 1500-wrapped cores C on $r_{\max}$ which do not serve as test partners of light-wrapped cores. Then the procedure searches through other TestRails to see whether there is sufficient idle time to fit in core i^*, whose test application time is the shortest in C (lines 8–17). For each TestRail r, the procedure searches through every idle range. Note, the idle time between r and $r_{\max}$ is also one of the idle ranges. If such a TestRail can be found, core i^* will be rescheduled on the new TestRail (line 14). The procedure exits when C is empty or no beneficial reassignment can be found.

For the example lightSOC depicted in Fig. 6, after step RescheduleBetweenRails, the IEEE 1500-wrapped core PC_5,

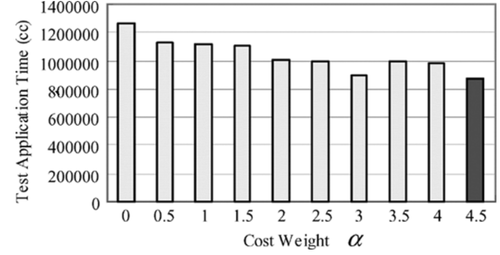

Fig. 12. Testing time variation for p34392 with different cost weight.

which originally sits on bottleneck TestRail3 and does not serve as test partner for any light-wrapped cores, is rescheduled to TestRail2, which reduces the overall SOC test application time, as shown in Fig. 11(b).

IV. Experimental Results

To analyze the effectiveness of the proposed solution on the overall test application time, experiments were carried out for three benchmark SOCs p22810, p34392, and p93791, originally provided from the ITC02 SOC *test benchmarking initiative* [27]. Since the functional interconnects are not provided in the benchmark files, we have randomly generated them to support the proposed approach as in [39]. For the generated interconnects, at most 12 cores in p22810, 8 cores in p34392 and 12 cores in p93791 can be light-wrapped.

First, for p34392 and $W_{\max} = 32$, we analyze the test application time variation with different values of the cost weight α (from 0 to 4.5), used by procedure BuildCostFunction. As shown in Fig. 12 that the variation can be large, which justifies the need to try different starting TestRail architectures for each of the $\mathrm{loopCnt} = 500$ iterations of the top-level Algorithm 1. These irregular variations of testing time are also observed for other TAM widths and other SOCs. The large number of trials for the initial architecture effectively compensates for the lack of optimization in the DesignTestRail step.

Tables II–IV show the test application time comparison between the proposed approach (T_{new}), the Test Bus-based approach presented in [39] $(T_{[39]})$, and the strategy that uses serial ExTest after the test of all the wrapped cores (T_s) using [15]. $\Delta T_{[39]}$ and ΔT_s are computed as $\Delta T_{[39]} = (T_{\mathrm{new}} - T_{[39]}/T_{[39]})$ and $\Delta T_s = (T_{\mathrm{new}} - T_s/T_s)$, respectively. Note that the first step of our algorithm (DesignTestRail) creates a configuration which is equivalent to the case where parallel ExTest with access to internal scan chains is applied after testing the wrapped cores. Consequently, because parallel ExTest (without access to internal scan chains) is a particular case of our method (i.e., the worst case scenario) the results for it are not reported. We have used four different light-wrapped core configurations for benchmark SOCs p22810, p34392 and p93791. For each SOC, when the number of light-wrapped cores (N_l) is increased we keep the light-wrapped cores from the previous experiments (with the lower number of light-wrapped cores). Note, the functional interconnects are generated randomly only once in this experiment in order to have a fixed interconnect topology for all the N_l values.

It can be observed that, in most cases ($W_{\max} \geq 8$ in this experiment), T_s is much higher than T_{new} and $T_{[39]}$, especially

TABLE II
TEST APPLICATION TIME COMPARISON FOR p22810 WITH DIFFERENT LIGHT-WRAPPED CORE CONFIGURATIONS

	SOC p22810									
	$N_l = 4$					$N_l = 6$				
W_{max}	T_{new} (cc)	$T_{[39]}$ (cc)	$\Delta T_{[39]}$ (%)	T_s (cc)	ΔT_s (%)	T_{new} (cc)	$T_{[39]}$ (cc)	$\Delta T_{[39]}$ (%)	T_s (cc)	ΔT_s (%)
4	1884702	3404904	-44.65	**1868106**	+0.89	**1923097**	3394368	-43.34	1951492	-1.46
8	**970189**	1229364	-21.08	1068491	-9.20	**1005906**	1224313	-17.84	1167562	-13.85
16	**493756**	589606	-16.26	614578	-19.66	**518821**	589606	-12.01	709055	-26.83
24	**339365**	403345	-15.86	459080	-26.08	**341626**	403345	-15.30	556665	-38.63
32	**251426**	305801	-17.78	380363	-33.90	**265770**	305801	-13.09	487274	-45.46
40	**207050**	241237	-14.17	340162	-39.13	**222032**	241237	-7.96	447073	-50.34
48	**177892**	197593	-9.97	321167	-44.61	**194200**	197593	-1.72	428078	-54.63
56	**151002**	174485	-13.46	295057	-48.82	**164127**	174485	-5.94	401968	-59.17
64	**151226**	174485	-13.33	295057	-48.75	**150862**	174485	-13.54	401968	-62.47
	$N_l = 8$					$N_l = 12$				
W_{max}	T_{new} (cc)	$T_{[39]}$ (cc)	$\Delta T_{[39]}$ (%)	T_s (cc)	ΔT_s (%)	T_{new} (cc)	$T_{[39]}$ (cc)	$\Delta T_{[39]}$ (%)	T_s (cc)	ΔT_s (%)
4	2058492	3382348	-39.14	**1948858**	+5.63	2136758	3382945	-36.84	**2018958**	+5.83
8	**1007307**	1224315	-17.72	1169454	-13.87	1304293	**1272316**	+2.51	1279673	+1.92
16	**569969**	589608	-3.33	792548	-28.08	739167	**724792**	+1.98	908568	-18.64
24	381566	**367262**	+3.89	662219	-42.38	578935	**444965**	+30.11	831719	-30.39
32	**282427**	330173	-14.46	589601	-52.10	405668	**326979**	+24.07	736484	-44.92
40	**232231**	241238	-3.73	589601	-60.61	332834	**269605**	+23.45	736484	-54.81
48	221089	**212920**	+3.84	552627	-59.99	335816	**259793**	+29.26	724159	-53.63
56	**183619**	191358	-4.04	525942	-65.09	276336	**223512**	+23.63	699510	-60.50
64	190899	**174486**	+9.41	525942	-63.70	258099	**182770**	+41.22	699510	-63.10

TABLE III
TEST APPLICATION TIME COMPARISON FOR p34392 WITH DIFFERENT LIGHT-WRAPPED CORE CONFIGURATIONS

	SOC p34392									
	$N_l = 2$					$N_l = 4$				
W_{max}	T_{new} (cc)	$T_{[39]}$ (cc)	$\Delta T_{[39]}$ (%)	T_s (cc)	ΔT_s (%)	T_{new} (cc)	$T_{[39]}$ (cc)	$\Delta T_{[39]}$ (%)	T_s (cc)	ΔT_s (%)
4	**4288630**	7276835	-41.06	4556299	-5.87	**4598506**	7276835	-36.81	4976584	-7.60
8	**2199507**	2781117	-20.91	2833300	-22.37	**2387461**	2781117	-14.15	3254232	-26.64
16	**1163432**	1396602	-16.70	1751894	-33.59	**1201762**	1396602	-13.95	2174364	-44.73
24	840216	**838643**	+1.88	1625768	-48.32	889207	**838643**	+6.03	2048238	-56.59
32	637660	**544579**	+17.09	1331704	-52.12	726572	**607678**	+19.57	1754174	-58.59
40	**544579**	**544579**	0	1331704	-59.11	571600	**570229**	+2.40	1754174	-67.41
48	**544579**	**544579**	0	1331704	-59.11	**544579**	**544579**	0	1754174	-68.96
56	**544579**	**544579**	0	1331704	-59.11	**544579**	**544579**	0	1754174	-68.96
64	**544579**	**544579**	0	1331704	-59.11	**544579**	**544579**	0	1754174	-68.96
	$N_l = 6$					$N_l = 8$				
W_{max}	T_{new} (cc)	$T_{[39]}$ (cc)	$\Delta T_{[39]}$ (%)	T_s (cc)	ΔT_s (%)	T_{new} (cc)	$T_{[39]}$ (cc)	$\Delta T_{[39]}$ (%)	T_s (cc)	ΔT_s (%)
4	**4636891**	7153222	-35.18	5671611	-18.24	**5276385**	7474887	-29.41	5741135	-8.10
8	**2384181**	2781117	-14.27	3981306	-40.12	**2923023**	3226091	-9.39	4110412	-28.89
16	**1324457**	1396602	-5.17	2915401	-54.57	1560145	**1396603**	+11.71	3310897	-52.88
24	**933378**	969392	-3.72	2813997	-66.83	1356128	**1243747**	+9.04	3249215	-58.26
32	**756336**	765643	-1.22	2519933	-69.99	1175547	**1119960**	+4.96	3049625	-61.45
40	726772	**572794**	+26.88	2519933	-71.16	1077662	**1036450**	+3.98	3012616	-64.23
48	622163	**544579**	+14.25	2519933	-75.31	1043252	**1013245**	+2.96	3012616	-65.37
56	605659	**544579**	+11.22	2519933	-75.97	1027602	**999723**	+2.79	3012616	-65.89
64	582992	**544579**	+7.05	2519933	-76.86	1028265	**996929**	+3.14	3012616	-65.87

when the TAM width is high. This is expected because the serial test of light-wrapped cores dominates the overall test application time with the increase of TAM width. However, for $W_{\max} = 4, T_s$ is close to or sometimes even better than T_{new}. This is mainly due to the following two reasons: (i) when the TAM width is small, the time for testing 1500-wrapped cores may dominate the entire SOC test application time; (ii) when the TAM width is very small, the flexible-width architecture used in [15] usually generates better schedules for 1500-wrapped cores than the fixed-width architecture used in the proposed methodology.

When the overall TAM width is small ($W_{\max} = 4, 8$ or 16 in this experiment), in almost all of the cases, the test application time for the proposed method is lower than the one reported in [39]. This is because, for the producer/CUT/consumer architecture from [39], separate producer and consumer TAMs have to be designed to shift in/out the test stimuli/responses to/from the producers and consumers of the light-wrapped cores. This leads to a decrease in the number of TAM resources for the CUT TAM group, which are used to shift in/out all the test data for IEEE 1500-wrapped cores and the internal scan chains in light-wrapped cores. In contrast, for the method proposed in this paper, and based on the *TestRail* architecture, all the TAM resources are used to load the test data for both IEEE 1500-wrapped cores and the internal scan chains of the light-wrapped cores. For [39], when $W_{\max}$ is small, the CUT TAM group dominates the test application time of the entire SOC, since its available bandwidth is low. When increasing $W_{\max}$, CUT TAM width no longer determines the bottleneck for the SOC test schedule, as it can be observed in Tables I, II, and III. Therefore, the improvements of the proposed solution disappear when increasing the $W_{\max}$ and we attribute the few contradictory cases to the fact that the fast heuristics cannot always lead to near-optimal results.

From Tables II–IV, it can also be seen that, when the number of light-wrapped cores N_l is small, the proposed method leads to

TABLE IV
TEST APPLICATION TIME COMPARISON FOR p93791 WITH DIFFERENT LIGHT-WRAPPED CORE CONFIGURATIONS

	SOC p93791									
	$N_l = 4$					$N_l = 6$				
W_{max}	T_{new} (cc)	$T_{[39]}$ (cc)	$\Delta T_{[39]}$ (%)	T_s (cc)	ΔT_s (%)	T_{new} (cc)	$T_{[39]}$ (cc)	$\Delta T_{[39]}$ (%)	T_s (cc)	ΔT_s (%)
4	**7915588**	13942991	-43.23	8933393	-11.39	**8395010**	13894497	-39.58	9560256	-12.19
8	**4048644**	4796178	-15.59	5464946	-25.92	**4333418**	4841393	-10.49	6447942	-32.79
16	**2105982**	2318569	-9.19	3711427	-43.26	**2246607**	2383685	-5.75	4874709	-53.91
24	**1382726**	1551868	-10.90	3075060	-55.03	**1562007**	1568659	-0.42	4299781	-63.67
32	**1035950**	1192928	-13.16	2894409	-64.21	**1201271**	1233571	-2.62	4125195	-70.88
40	**858994**	980098	-12.36	2710920	-68.31	**951174**	974074	-2.35	3999004	-76.21
48	**736656**	809219	-8.97	2481680	-70.32	868083	**841379**	+3.17	3768757	-76.97
56	635909	**628907**	+1.11	2439721	-73.94	**683161**	696490	-1.91	3726798	-81.67
64	**548069**	592361	-7.48	2385542	-77.03	623143	**580302**	+7.38	3672619	-83.03
	$N_l = 8$					$N_l = 12$				
W_{max}	T_{new} (cc)	$T_{[39]}$ (cc)	$\Delta T_{[39]}$ (%)	T_s (cc)	ΔT_s (%)	T_{new} (cc)	$T_{[39]}$ (cc)	$\Delta T_{[39]}$ (%)	T_s (cc)	ΔT_s (%)
4	**8747510**	13876650	-36.96	9739362	-10.18	**9924700**	13831009	-28.24	13717481	-27.65
8	**4567028**	4839660	-5.63	6724752	-32.09	5174018	**5072044**	+2.01	10736258	-51.81
16	**2366156**	2383686	-0.74	5271798	-55.12	2879062	**2566209**	+12.19	9397331	-69.36
24	1647152	**1594786**	+3.28	4737040	-65.23	1969220	**1803658**	+9.18	8934856	-77.96
32	1264935	**1233572**	+2.54	4564989	-72.29	1515097	**1278032**	+18.55	8764197	-82.71
40	1046972	**937510**	+11.68	4447378	-76.46	1206425	**1128936**	+6.86	8607313	-85.98
48	844862	**841379**	+0.41	4248526	-80.11	1089628	**937252**	+16.26	8518860	-87.21
56	786894	**714528**	+10.13	4206567	-81.29	945671	**749536**	+26.17	8476901	-88.84
64	656602	**616617**	+6.48	4152388	-84.19	898136	**678392**	+32.29	8422722	-89.34

shorter test application time, while [39] gives better results when N_l is large. This is because, again, in the TestRail-based method presented in this paper, there are no dedicated producer and consumer TAM groups. When N_l is small, the test data to be shifted for the light-wrapped cores is small. In this case, the load time for all the light-wrapped cores that are scheduled at the same time is low and thus the testing time is dominated by the time needed to load data in the internal scan chains. When N_l is large, whenever a light-wrapped core is scheduled, it is very likely that the schedule of several other concurrently tested light-wrapped cores will be affected (and hence the TAM resources cannot be used to test other cores). In addition, when considering all of the loading time for all the test partners for all the light-wrapped cores scheduled at the same time, the test application time for the overlapped patterns is significantly increased (see Example 1). For [39], since dedicated producer and consumer TAMs are used to shift in/out the test data for the light-wrapped cores, the CUT TAM resources can be fully exploited to test the wrapped cores, and hence [39] is more efficient when N_l is large. For SOC p34392 when $W_{\max} = 32$, the result obtained in [39] is better even when $N_l = 2$ or $N_l = 4$. This is because, in these two cases, core 18, which is quite large and dominates the test application time of the entire SOC, is IEEE 1500-wrapped. If the proposed method places the schedule of the light-wrapped cores in front of core 18, the test application time will further increase.

V. CONCLUSION

This paper has discussed our investigation into the reuse of the TestRail architecture for rapid and concurrent test of wrapped cores and unwrapped logic blocks using only the test control mechanism and the test modes and instructions available through IEEE SECT. It was found that, when the available number of test pins or tester channels and/or the number of unwrapped logic blocks are small, the test scheduling algorithm proposed in this paper outperforms the previous solution based on extending the Test Bus architecture [39]. The method proposed in this paper is particularly suitable when rapid testing is an objective and a small number of logic blocks (either embedded cores or user defined logic) cannot be wrapped due to methodology constraints, timing violations or area constraints. In addition, the proposed approach is preferable when the SOC design has a constrained number of I/O pins available for test or when tester channels are limited. In summary, because from the practical standpoint usually only a limited number of unwrapped logic blocks exist in a complex SOC, and at the same time multisite testing is gaining acceptance to reduce SOC test cost in the industry, the proposed solution can improve the time the chip spends on the tester.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their constructive comments.

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.* , Washington, DC, Oct. 1998, pp. 448–457.

[2] R. W. Bassett, B. J. Butkus, S. R. Dingle, M. R. Faucher, P. Gillis, J. H. Panner, J. J. G. Petrovick, and D. L. Wheater, "Low-cost testing of high-density logic components," *IEEE Design Test Computers*, vol. 7, no. 2, pp. 15–28, Apr. 1990.

[3] K. Chakrabarty, "Design of system-on-a-chip test access architectures using integer linear programming," in *Proc. IEEE VLSI Test Symp.*, Montreal, QC, Canada, Apr. 2000, pp. 127–134.

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

[5] 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 Integr. Circuits Syst.*, vol. 19, no. 8, pp. 863–877, Aug. 2000.

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

[7] ——, "On-chip test infrastructure design for optimal multi-site testing of system chips," in *Proc. Design, Automation, and Test in Europe (DATE)*, Munich, Germany, Mar. 2005, pp. 44–49.

[8] P. Harrod, "Testing reusable IP—A case study," in *Proc. IEEE Int. Test Conf.*, Atlantic City, NJ, Sept. 1999, pp. 493–498.

[9] Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, and S. M. Reddy, "Static pin mapping and SOC test scheduling for cores with multiple test sets," in *Proc. Int. Symp. Quality of Electronic Design*, Mar. 2003, pp. 99–104.

[10] 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.*, Kyoto, Japan, Nov. 2001, pp. 265–270.

[11] Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, O. Sammon, and S. M. Reddy, "On concurrent test of core-based SOC design," *J. Electron. Testing: Theory Applicat.*, vol. 18, no. 4/5, pp. 401–414, Aug. 2002.

[12] *IEEE Standard for Embedded Core Test—IEEE Std. 1500–2004. IEEE*, 2004.

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

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

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

[16] ——, "Efficient test access mechanism optimization for system-on-chip," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 22, no. 5, pp. 635–643, May 2003.

[17] ——, "Test access mechanism optimization, test scheduling, and tester data volume reduction for system-on-chip," *IEEE Trans. Comput.*, vol. 52, no. 12, pp. 1619–1632, Dec. 2003.

[18] V. Iyengar, S. K. Goel, K. Chakrabarty, and E. J. Marinissen, "Test resource optimization for multi-site testing of SOC's under ATE memory depth constraints," in *Proc. IEEE Int. Test Conf.*, Baltimore, MD, Oct. 2002, pp. 1159–1168.

[19] R. Kapur and T. W. Williams, "Manufacturing test of SoCs," in *Proc. IEEE Asian Test Symp.*, Tamuning, Guam, Nov. 2002, pp. 317–319.

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

[21] S. Koranne and V. Iyengar, "On the use of k-tuples for SoC test schedule representation," in *Proc. IEEE Int. Test Conf.* , Baltimore, MD, Oct. 2002, pp. 539–548.

[22] 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.*, 2001, pp. 59–64.

[23] E. Larsson and H. Fujiwara, "Test resource partitioning and optimization for SOC designs," in *Proc. IEEE VLSI Test Symp.*, Napa, CA, Apr. 2003, pp. 319–324.

[24] E. Larsson and Z. Peng, "An integrated framework for the design and optimization of SOC test solutions," *J. Electron. Testing: Theory Applicat.*, vol. 18, no. 4/5, pp. 385–400, Aug. 2002.

[25] ——, "A reconfigurable power-conscious core wrapper and its application to SOC test scheduling," in *Proc. IEEE Int. Test Conf.*, Charlotte, NC, Sep. 2003, pp. 1135–1144.

[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," in *Proc. IEEE Int. Test Conf.*, Washington, DC, Oct. 1998, pp. 284–293.

[27] E. J. Marinissen, V. Iyengar, and K. Chakrabarty, ITC'02 SOC Test Benchmarks.

[28] 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. Testing: Theory Applicat.*, vol. 18, no. 4/5, pp. 365–383, Aug. 2002.

[29] M. Nourani and C. Papachristou, "Structural fault testing of embedded cores using pipelining," *J. Electron. Testing: Theory Applicat.*, vol. 15, no. 1, pp. 129–144, 1999.

[30] B. Pouya and N. Touba, "Modifying user-defined logic for test access to embedded cores," in *Proc. IEEE Int. Test Conf.*, Washington, DC, Nov. 1997, pp. 60–68.

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

[32] A. Sehgal, V. Iyengar, M. D. Krasniewski, and K. Chakrabarty, "Test cost reduction for SOC's using virtual TAM's and lagrange multipliers," in *Proc. ACM/IEEE Design Automation Conf.*, Anaheim, CA, Jun. 2003, pp. 738–743.

[33] C.-P. Su and C.-W. Wu, "A graph-based approach to power-constrained SOC test scheduling," *J. Electron. Testing: Theory Applicat.*, vol. 20, no. 1, pp. 45–60, Feb. 2004.

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

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

[36] B. Vermeulen, S. Oostdijk, and F. Bouwman, "Test and debug strategy of the PNX8525 nexperia™ digital video platform system chip," in *Proc. IEEE Int. Test Conf.*, Baltimore, MD, Oct. 2001, pp. 121–130.

[37] E. H. Volkerink, A. Khoche, J. Rivoir, and K. D. Hilliges, "Test economics for multi-site test with modern cost reduction techniques," in *Proc. IEEE VLSI Test Symp.*, 2002, pp. 411–416.

[38] H. Vranken, T. Waayers, H. Fleury, and D. Lelouvier, "Enhanced reduced-pin-count test for full-scan design," in *Proc. IEEE Int. Test Conf.*, Oct. 2001, pp. 738–747.

[39] Q. Xu and N. Nicolici, "On reducing wrapper boundary register cells in modular SOC testing," in *Proc. IEEE Int. Test Conf.*, Charlotte, NC, Sep. 2003, pp. 622–631.

[40] ——, "Multi-frequency test access mechanism design for modular SOC testing," in *Proc. IEEE Asian Test Symp.*, Kenting, Taiwan, R.O.C., Nov. 2004, pp. 2–7.

[41] ——, "Resource-constrained system-on-a-chip test: A survey," in *Proc. Inst. Elect. Eng. Comput. Digit. Tech.*, vol. 152, Jan. 2005, pp. 67–81.

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

[43] D. Zhao and S. Upadhyaya, "Power constrained test scheduling with dynamically varied TAM," in *Proc. IEEE VLSI Test Symp.*, 2003, pp. 273–278.

[44] 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.

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

Qiang Xu (S'03) received the B.E. and M.E. degrees in telecommunication engineering from 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.

During 2000–2001, he was with a start-up integrated circuit design house. He is currently an Assistant Professor of computer science and engineering with 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.

Dr. Xu was the recipient of the Best Paper Award for the 2004 IEEE/ACM Design, Automation and Test in Europe (DATE) Conference and Exhibition.

Nicola Nicolici (S'99–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 currently an Assistant Professor of computer engineering with McMaster University, Hamilton, ON, Canada. His research interests are in the area of computer-aided design and test, and he has authored a number of papers in this area.

Dr. Nicolici is a member of ACM SIGDA and the IEEE Computer and Circuits and Systems Societies. He was the recipient of 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 serves on the editorial board of *IEE Proceedings—Computers and Digital Techniques*.