

On Reducing Wrapper Boundary Register Cells in Modular SOC Testing

Qiang Xu and Nicola Nicolici

Department of Electrical and Computer Engineering
McMaster University, Hamilton, ON L8S 4K1, Canada
Email: xuqiang@grads.ece.mcmaster.ca, nicola@ece.mcmaster.ca

Abstract

Motivated by the increasing area and performance overhead caused by wrapping the embedded cores for modular SOC testing, this paper proposes a solution for reducing the number of wrapper boundary register cells. Since the very purpose of core wrappers is to provide controllability and observability for the cores-under-test, it is shown how the number of wrapper boundary register cells can be reduced without affecting the test quality. While a testing time overhead, caused by lower test concurrency, is incurred, there are clear benefits in reducing the necessary DFT area and especially in decreasing the propagation delays, which can improve the SOC's functional timing performance.

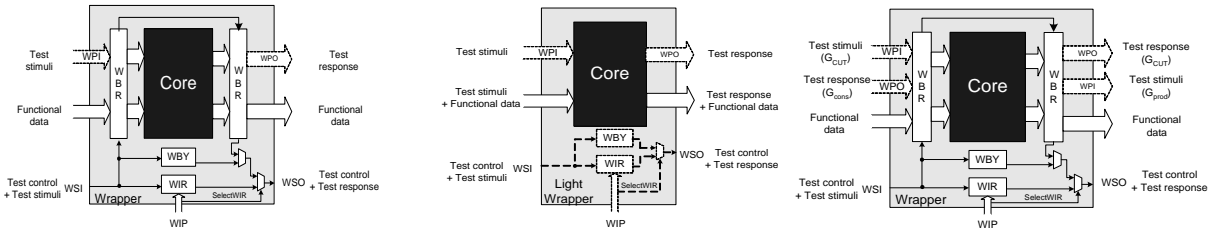
1 Introduction

Semiconductor manufacturing technology has advanced significantly over the past couple of decades and continues to do so. The International Technology Roadmap for Semiconductors (ITRS) [5] projects that multi-billion transistor chips will be manufactured by the end of this decade and new implementation methodologies are emerging to handle with chip complexity. For example, 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. The IP cores are pre-designed and pre-verified by the core providers, however SOC composition is the system integrators' duty, who is also in charge of verification and manufacturing testing of the entire SOC, including the IP-protected internal cores.

The IP core reuse reduces the design cycle, however the rapid increase in SOC complexity makes the test development a major implementation bottleneck [15]. 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. Therefore, special test access mechanisms (TAMs) are required to facilitate core-based SOC test. To enable both core reuse and easy test access, the embedded cores are connected to the TAMs using special interfaces called core wrappers [13, 15]. While

the use of core wrappers guarantees high test quality, by facilitating the test each core individually, the design for test (DFT) area overhead associated with all the wrappers may adversely influence the cost of the test. Moreover, since both core's inputs and outputs are buffered, two sets of multiplexers (one for the inputs and one for the outputs) are required to switch between the functional and test mode 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 functional timing 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 [14]. Despite avoiding the high number of multiplexers, the main limitation of the solution presented in [14] 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 scalability. 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 pre-analyzed test set.

To lower the DFT area and performance overhead by reducing the number of wrapper cells, the main challenge lies in finding a solution which is compatible with the P1500 standard [13] and, at the same time, preserve the modularity and scalability of the existing tool flows for TAM design and test scheduling [3, 4, 6, 8, 10]. 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 wrapper boundary register cells are removed. By reusing the functional interconnect topology, new scalable and modular algorithms for TAM design and test scheduling are proposed. It is shown through experimental results that up to half of the cores can have the wrapper boundary register cells removed (the numbers are dependent on the interconnect topology) without affecting the test quality.



(a) Typical IEEE P1500 Wrapper

(b) Light Wrapper Without WBR

(c) Revised IEEE P1500-Compliant Wrapper for Producer/Consumer Cores

Figure 1. Wrapper Architectures.

2 Light Wrappers for Embedded Cores

This section provides an overview of core-based SOC testing and introduces a new concept called *light wrappers* and explains how these light wrappers can address area and performance overhead.

IEEE P1500 Standard for Embedded Core Test (SECT) is an ongoing standard that intends to facilitate core-based testing [13]; its architecture consists of test control lines, test access mechanism and core wrappers. A typical P1500 wrapper is shown in Figure 1(a); the test control lines set the mode of each core and the test access mechanism is used to transfer data to and from the core-under-test (CUT). The wrapper boundary register (WBR) cells not only provide a core isolation mechanism used by the test modes (e.g., INTEST or EXTEST), but also provide full controllability and observability to all the core terminals. In the INTEST mode, used for testing the core's internal logic, the input WBR cells act as primary inputs (PIs) to the CUT, while the output WBR cells act as primary outputs (POs). In the EXTEST mode, when all the cores are wrapped, the goal is to test the interconnect wires between the cores. Thus the output WBR cells provide patterns and the input WBR cells capture responses from the interconnect, which are blocked in the input WBR cells and do not get propagated to the core's internal logic.

From the system integrator's standpoint, to test the embedded cores and its interconnect, full controllability and observability needs to be provided at the input and output of each core (note, to ensure modularity and scalability, the controllability and observability 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 $Core_i$, 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. In Figure 2, $Core_3$ is not wrapped with WBR cells, however all its producers ($Core_1, Core_2$) and its consumer ($Core_4$) are wrapped. For INTEST of $Core_3$, the controllability of its input terminals is provided through its producers' outputs

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 cells of $Core_4$. Because 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 $Core_3$. In this model a P1500-compliant core can serve as a producer and consumer at the same time. $Core_3$ has a light wrapper (Figure 1(b)) and the producer and consumer cores need to be revised to support the previously explained transfer mechanisms (see Figure 1(c) whose parameters are detailed in Section 3.2).

To summarize the above-explained observation, a core can be tested without wrapping its terminals as long as all its producers and consumers are wrapped. If the cores do not comprise any scan chains then they do not need a wrapper at all (Figure 1(b)), however the producers and consumers must be updated with a P1500-compliant wrapper (Figure 1(c)). If the cores are P1500-ready and include internal scan chains, then, in addition to updating the producers and consumers wrappers (Figure 1(c)) the tested core needs a light wrapper (Figure 1(b)), which will not affect the performance. This light wrapper requires either WSI/WSO or WPI/WPO to shift in the test stimuli and shift out the test response for the internal scan chains; the light wrapper also includes serial or parallel bypass register (WBY) to enable a shortened test access path to other cores. In addition, the light wrapper comprises the WIP to control the test mode. From now onwards *light-wrapped cores* will refer to cores which do not need a wrapper at all or cores with a light wrapper, since both of them reduce performance overhead.

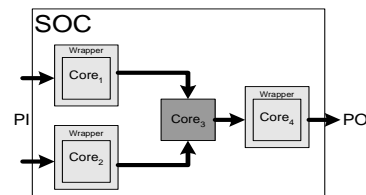


Figure 2. Full Controllability and Observability for $Core_3$ Without WBR Cells.

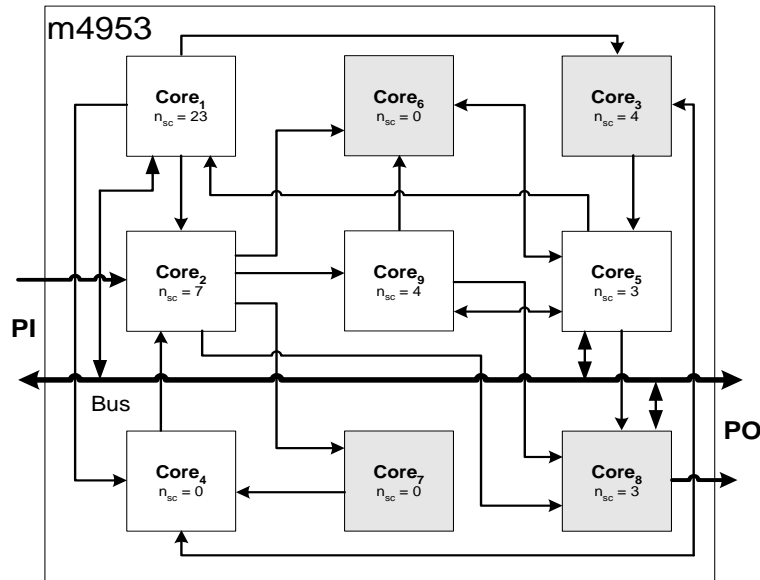


Figure 3. Example SOC: m4953.

3 TAM Design and Test Scheduling

Having introduced the light wrapper concept and outlined its applicability to P1500-based testing, this section focuses on new TAM design and test scheduling issues which arise from the usage of light-wrapped cores.

To clarify all the issues related to testing these light-wrapped cores, we provide a hypothetical SOC m4953. This hypothetical SOC has nine cores inside and there is also a system bus connecting 3 cores. The number of scan chains and the functional interconnects of these cores are shown in Figure 3¹. The name of this benchmark circuit follows the benchmark naming convention presented in [12], where *m* refers to McMaster University and the number 4953 denotes the test complexity.

3.1 Test Conflicts Caused by Sharing Functional Interconnect and Producers/Consumers

It is known that in the INTEST mode [3], all the P1500 wrapped cores can be tested concurrently, as long as they use different TAM lines. However, because testing light-wrapped cores is dependent on its producers and consumers, TAM lines conflicts are not the only ones which limit the test concurrency. To avoid loss in test quality or increased control and/or computational complexity, there are five new types of test conflicts, described as follows:

- **Producer-CUT Conflict:** Producer(s) and the CUT should **not** be tested at the same time. For example, in Figure 3, if *Core₆* is a light-wrapped core, *Core₂*, *Core₅* and *Core₉* should not be tested at the same time as *Core₆*. This is because, the producer needs to utilize

its output WBR to capture its test response, however, at the same time, the CUT needs the producer's output WBR to provide test stimuli. If they are tested concurrently the test data will be corrupted.

- **CUT-Consumer Conflict:** The CUT and consumer(s) should **not** be tested at the same time. For example, in Figure 3, if *Core₂* is a light-wrapped core, *Core₆*, *Core₇*, *Core₈* and *Core₉* should not be tested at the same time. This is because, the consumer needs to utilize its input WBR to deliver test stimuli, however, at the same time, the CUT needs the consumer's input WBR to capture the test response. If they are tested concurrently the test data will be corrupted.
- **Shared-Producer Conflict:** Two light-wrapped cores which connect directly (i.e., on a dedicated non-shared set of lines) to the same producer **cannot** be tested at the same time. For m4953, if *Core₇* and *Core₈* are light-wrapped cores, they cannot be tested at the same time because they are connected directly to the same producer *Core₂* and both of them require the output WBR of *Core₂* to provide the test stimuli.
- **Shared-Consumer Conflict:** Two light-wrapped cores which connect directly to the same consumer **cannot** be tested at the same time. For example, if *Core₃* and *Core₆* are light-wrapped cores, they cannot be tested at the same time since they connect directly to the same consumer *Core₅*, and hence both of them need the input WBR of *Core₅* to capture the test responses.
- **Shared-Bus Conflict:** If the producer(s) or consumer(s) connect to the light-wrapped CUT through functional buses, they **may not** be tested at the same time. For

¹to make the drawing clear, the cores are not placed in the increasing numerical order; additional test set parameters for the cores can be found at <http://www.ece.mcmaster.ca/~nicola/m4953.soc>

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 WBR 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.

3.2 TAM Division Into Three Groups: Producers, Cores-Under-Test and Consumers

The previous section has outlined the test conflicts which, if not taken into consideration, may reduce the test quality or increased control and/or computational complexity. Other types of conflicts may appear if the test data is transferred using shared TAM lines between producers, cores under test and consumers. To avoid this type of conflicts, which may adversely influence the testing time, dividing the TAM lines into three groups is proposed, motivated by the following examples:

Example 1 Consider the benchmark circuit m4953 shown in Figure 3 and let's suppose 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. Assume that $Core_1$ and $Core_2$ share the same TAM lines (TAM_{w1}) and $Core_6$ connects to a different TAM group (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 separately all the light-wrapped cores, thus leading to increased testing time.

Example 2 Shared TAM loading can also increase the test control complexity. For example, in the case of m4953 shown in Figure 3, 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 the pattern 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, consumers and CUTs, all of these operations introduce additional synchronization issues and consequently they may increase the test control complexity.

If there are only a very small number n ($n \leq 2$) of light-wrapped cores in the SOC, then using WSI/WSO to load/unload the test stimuli/response may be a neat solution. After testing all the P1500-compliant cores, we can test the n light-wrapped cores serially by putting its producers and consumers into the EXTEST mode. However, if the number of light-wrapped cores is larger, then the one-bit test access mechanism provides limited bandwidth and hence it will considerably increase the testing time.

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 separately. This division, will remove additional test conflicts. Using the setup from Example 1, testing $Core_6$ will need the assistance of $Core_2$ to provide the test stimuli. If the output WBR of $Core_2$ is loaded through G_{prod} and, although $Core_1$ and $Core_2$ share the same TAM resources in G_{CUT} , $Core_1$ can still be tested at the same time as $Core_6$. To speed up the loading time, the G_{prod} needs to connect only to the output WBRs of the cores which serve as producers during testing; similarly G_{cons} needs to connect only to the input WBRs of the cores which serve as consumers (see Figure 1(c)). As a result, we need to introduce two new wrapper instructions: *LOADPROD* for the producers and *UNLOADCONS* for the consumers.

The multiplexing and daisychain architectures [1] are not suitable for G_{CUT} assignment. For example, if a core was provided with only one long scan chain and we have five TAM lines for G_{CUT} , it is a waste of resources to assign all the five TAM lines to this core. Hence, G_{CUT} group uses a test bus architecture (a combination of the multiplexing and distribution architectures [1]). However, for G_{prod} and G_{cons} we use the multiplexing architecture [1]. In this multiplexing architecture, all the producers/consumers connect to all G_{prod}/G_{cons} lines directly and get full access to the TAM resources; additional multiplexers select which producer/consumer cores are connected to the output pins. Note, in the daisychain architecture, all the TAM lines construct long scan chains over all the modules and also get full access to the TAM resources. However, since the daisychain architecture depends on by-passes to shorten the time to access each core, we have to change the test mode of the producers/consumers with *BYPASS/LOADPROD/UNLOADCONS* instructions for all the related cores, thus increasing the test control complexity. An additional reason for using the multiplexing architecture for G_{prod}/G_{cons} is that it can almost always give a optimal loading/unloading time for a given width W_{prod}/W_{cons} . If the number of the outputs of its producers is N_o , then the loading time will be $\lceil \frac{N_o}{W_{prod}} \rceil$. As long as $W_{prod} \leq N_o$ (which is realistic in most of the cases), there is no waste for G_{prod} TAM resources, which leads to an optimal loading time for its producers. The same holds for G_{cons} unloading.

Another interesting feature of the proposed solution lies in computing the testing time associated with each light-wrapped core. When one light-wrapped core is tested at the same time with other light-wrapped cores, because the producers and consumers have to be loaded serially using G_{prod} and G_{cons} , which are implemented using a multiplexing architecture, its *testing time is not dependent only on its own producers, consumers and internal scan chain loading/unloading time*; rather it also depends on the producers (consumers) loading (unloading) times for all the light-wrapped cores tested at the same time.

Example 3 For $m4953$ shown in Figure 3, let's assume that $Core_3$ and $Core_8$ are light-wrapped cores and they are scheduled to be tested at the same time. Let's consider that the loading time for $Core_3$'s producers, consumers and internal scan chains are 10, 10 and 10 clock cycles respectively. Let's assume that the loading time for $Core_8$'s producers, consumers and internal scan chains are 20, 10 and 10 clock cycles respectively. Since the producers and consumers are connected using a multiplexing architecture [1], in order to apply a test pattern for $Core_3$, we must wait for the completion of the loading time for the producers of $Core_8$ as well. Therefore, one needs to wait for $10 + 20 = 30$ clock cycles to apply a test pattern to $Core_3$. If $Core_8$ is substituted by $Core_7$ with its producers loading time equal to 15 clock cycles, then one new pattern can be applied every $10 + 15 = 25$ clock cycles.

To deal with situation described in the previous example and to keep the control and computational complexity low, we propose to *align* test patterns for all the concurrently-tested light-wrapped cores. That is, if two light-wrapped cores are scheduled at the same time, then test patterns for both cores will have the same start times. Obviously, the core which needs the least time to load/unload its test stimuli/response will wait for all the currently scheduled cores to complete loading/unloading, thus leading to an increase in testing time.

3.3 Proposed Algorithms for Wrapper/TAM Co-optimization

The introduction of producers, consumers and TAM division into three groups, leads to new algorithms for wrapper/TAM co-optimization, as explained in this section.

Problem P_{WT-opt} : Given the test set parameters for each core, the total TAM width W 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 a wrapper design for each core, and a test schedule for the entire SOC such that: (i) the wrapper design constraints C_w are satisfied; (ii) the total number of light-wrapped cores is maximized; (iii) the total number of TAM lines used at any time does not exceed W ; and (iv) the overall SOC testing time is minimized.

Algorithm 1 - TAM_Division_And_Schedule

INPUT: $C, R, W, weight$

OUTPUT: $W_{prod}, W_{CUT}, W_{cons}$

$wrapper_type, schedule, testing_time$

```

1.  $wrapper\_type = Decide\_Wrapper\_Type(C, R);$ 
2.  $TIG = Construct\_TIG(wrapper\_type, R);$ 
3. For  $W_{CUT}$  from  $W - 2$  downto 1 {
4.    $W_{prod\_plus\_cons} = W - W_{CUT};$ 
5.   For  $W_{prod}$  from 1 to  $W_{prod\_plus\_cons} - 1$  {
6.      $W_{cons} = W_{prod\_plus\_cons} - W_{prod};$ 
7.      $testing\_time = Adapted\_TAM\_Schedule\_Optimizer($ 
.        $C, TIG, W_{prod}, W_{CUT}, W_{cons});$ 
8.      $localmin = \min\{all\ testing\_time\};$ 
9.     Record  $W_{prod\_localmin}, W_{cons\_localmin};$ 
.   }
10. if ( $localmin > weight \times globalmin$ )
11.   break; /*Prune search space*/
12.  $globalmin = \min\{all\ localmin\};$ 
13. Record  $W_{prod\_globalmin}, W_{cons\_globalmin};$ 
. }
14.  $W_{prod} = W_{prod\_globalmin};$ 
.  $W_{cons} = W_{cons\_globalmin};$ 
.  $W_{CUT} = W - W_{prod} - W_{cons};$ 
15. return  $W_{prod}, W_{CUT}, W_{cons}, wrapper\_type,$ 
.    $schedule, testing\_time;$ 

```

There are mainly two types of wrapper design constraints C_w : (i) if the critical paths appear between cores then, to avoid performance penalty, some cores must be light-wrapped; (ii) if some of the cores are provided with P1500 wrappers and their overhead does not affect the performance of the SOC, then there is no reason to make them light-wrapped. The proposed algorithm finds the optimal TAM division for each TAM group through enumeration. Note, the optimal TAM division refers to the best combination of W_{prod} , W_{CUT} and W_{cons} , which gives the minimum testing time for the entire SOC.

The enumerative algorithm to solve P_{WT-opt} is shown in Algorithm 1. The inputs are the set of cores (C), TAM width (W), functional interconnect relationship between cores (R) and a weight parameter ($weight$), used in pruning the search space. The outputs of the **TAM_Division_And_Schedule** algorithm are the number of TAM lines allocated to each group, wrapper type and design for each core, SOC test schedule and the minimum value for the testing time. The first decision determines which cores are to be light-wrapped according to the functional interconnect relationship R of the SOC (line 1). Based on the wrapper type (light-wrapped or not) and the test conflicts determined by functional interconnect relationship between cores (R), a Test Incompatibility Graph (TIG) is created (line 2). Next, the algorithm will enumeratively find the optimal TAM di-

vision and the system testing time *globalmin*. In the inner loop of this algorithm (lines 5 to 9), the local minimum testing time *localmin* for a fixed total width of $W_{prod} + W_{cons}$ ($W_{prod+plus-cons}$) 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 a *Quasi-convex* function with respect to W_{CUT} (i.e., parts of the function are convex). This means that it keeps decreasing until it reaches a local minimum value, at which point it starts increasing. This can be explained by the fact that when W_{CUT} has a small value, the testing time is dominated by the time to transfer test data through G_{CUT} (for justification see Equations (1) and (2) in the following section), and increasing W_{CUT} will finally break this bottleneck. The testing time starts to increase when the time required to load/unload the producer/consumer output/input WBR cells starts to dominate the test access time. There are some variations around the local minimum value, which can be justified by the dynamic rectangle packing (explained later in this section) caused by the multiplexing architecture used for G_{prod} and G_{cons} . 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). In our experiments we have found that *weight* = 1.1 is sufficient to obtain an optimal division (lines 10 and 11).

It should be noted that during the enumeration process, we do not need to do TAM design for G_{prod}/G_{cons} groups, since the connection between producers/consumers is fixed using the multiplexing architecture [1]. To obtain a TAM design and a test schedule for G_{CUT} , we adapt and use an existing rectangle packing algorithm *TAM.Schedule.Optimizer* [8]. In test scheduling the core tests are represented as rectangles [9] (the height of the rectangle represents the TAM width assigned to the core and the width of the rectangle represents the core's testing time). The rectangles are packed into a bin of fixed height (the total TAM width W), however due to the usage of a multiplexing architecture for producers/consumers, a *dynamic adaptation* of the existing algorithms is necessary. We elaborate on each of the main steps of the algorithm in the following paragraphs.

Decide Wrapper Type: Not all the cores need to be wrapped in an SOC, however, to provide full controllability and observability for the light-wrapped cores, they need to be surrounded by P1500-compliant cores and all its producers and consumers must be wrapped with WBR cells. The pseudocode for deciding the wrapper type is shown in Algorithm 2. The algorithm takes the set of cores C and the functional interconnect relationship R as the inputs, and it outputs the wrapper type for each core $i \in C$. First,

Algorithm 2 - Decide_Wrapper_Type

INPUT: C, R

OUTPUT: *wrapper_type*

```

/* According to direct functional interconnects */
1. For each Core  $i \in C$  {
2.   if (wrapper design constraints exist) {
3.     Wrap core  $i$  according to its constraint
4.   } else {
5.     Set Is_Light_Wrappedi = true;
6.     Initialize test_dependencyi;
7.   }
8. }
9. While (test_dependencyi != 0 for any core  $i \in C$ ) {
10.  Find Core  $j$  with the maximum test_dependency;
11.  Set Is_Light_Wrappedj = false; test_dependencyj = 0;
12.  Update test_dependency of its producers and consumers;
13. }
/* According to functional bus interconnects */
14. For each functional bus
15.  if (No core on the bus is wrapped)
16.    Wrap the core with the least number of I/Os;
17. return wrapper_type for each core;

```

the cores which need to be wrapped by P1500-compliant wrappers according to direct functional relationship (i.e., dedicated non-shared 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 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 a large number of P1500-compliant cores to test it. For example, in the case of m4953, *Core₂* has 2 producers *Core₁* and *Core₄*, and 4 consumers *Core₆*, *Core₇*, *Core₈*, *Core₉*, hence its *test_dependency* is initialized to 6. If *Core₂* is a light-wrapped core, we need to wrap all its 6 neighbors, which may lead to large DFT area and performance overhead. As a result, it is better to wrap *Core₂* with a P1500-compliant wrapper. Therefore, the algorithm finds those 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 as 0, since it does not require any producers and consumers; the *test_dependency* of all its unwrapped 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

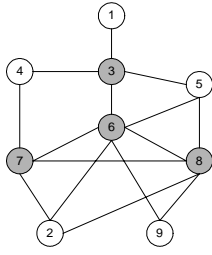


Figure 4. m4953 Test Incompatibility Graph.

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 since this will decrease the time required to load/unload the test stimuli/response. To illustrate the outcome of the proposed algorithm, in the case of m4953, *Core₃*, *Core₆*, *Core₇* and *Core₈* are selected to be light-wrapped (if no wrapper design constraints exist), as shown by the shaded boxes in Figure 3. Note, unlike in boundary scan-based testing, where chips are manufactured before the board is assembled, in core-based SOC testing the system integrator has this flexibility of light-wrapping cores without sacrificing controllability and observability.

Construct the Test Incompatibility Graph (TIG): If there are test conflicts between two cores (see Section 3.1), then these two cores are incompatible. We construct a test incompatibility graph *TIG* by treating each core as a node and by adding an edge between two nodes if the cores are incompatible. This *TIG* is used in Algorithm 3 (*Adapted_TAM_Schedule_Optimizer*). The *TIG* generated for m4953 is shown in Figure 4. As shown in the figure, edges illustrating incompatibility can exist only between two light-wrapped cores or between light-wrapped cores and its producers and consumers.

Dynamic Rectangle Representation: For a P1500-compliant core, if the assigned TAM width in G_{CUT} is given, the testing time to apply the entire test set T_p is determined by the Equation 1, 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 a best fit decreasing (BFD) algorithm [7] for wrapper design, $s_i(s_o)$ has a fixed value for a given TAM width, and hence the core test can be represented as a *static* rectangle. However, for a light-wrapped core, if the assigned TAM width is given, its testing time is determined by the Equation 2.

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

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

As described earlier in Example 3, when reusing functional interconnect for test data transfers, T_l does not only depend on the time to load its own producers (L_{prod}), consumers (L_{cons}) and internal scan chains (L_{in}); the testing

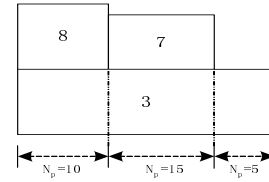


Figure 5. Loading Time for Different Patterns.

time also depends on the time necessary to load/unload all the concurrently-tested light-wrapped cores' producers/consumers. If for the given core the test schedule changes s times, then for each subset of patterns p_s (for the s distinct divisions of the time allocated to the given core) the testing time will be computed based on the light-wrapped cores scheduled in each of these s divisions. Note also that by using $\max\{\sum L_{prod}, \sum L_{cons}, \max\{L_{in}\}\}$, we align the start times of each test pattern of all the light-wrapped cores tested concurrently.

Example 4 *In the case of m4953, *Core₃* is compatible to light-wrapped cores *Core₇* and *Core₈*. Let's assume *Core₇* and *Core₈* are selected to be scheduled at the same test time with *Core₃*, as shown in Figure 5 (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 *Core₃* is updated each time the schedule changes. For the first 10 patterns, the loading time for both *Core₃* and *Core₈* will be $\max\{L_{prod,3} + L_{prod,8}, L_{cons,3} + L_{cons,8}, L_{in,3}, L_{in,8}\}$. However, for the next 15 patterns, once the test for *Core₈* has been completed and *Core₇* is scheduled concurrently with *Core₃*, the loading time will be $\max\{L_{prod,3} + L_{prod,7}, L_{cons,3} + L_{cons,7}, L_{in,3}, L_{in,7}\}$. The same reasoning is applied for the last 5 patterns when *Core₃* is not concurrent with any other light-wrapped cores. The variations in the loading time for each of the 3 divisions of the schedule for *Core₃* can be differentiated using a *loadSize* value determined by all the concurrently-tested light wrapped cores.*

Since testing time T_l for a light-wrapped core may change with its schedule, this *dynamic* attribute leads to *dynamic rectangle representation* of the core's test. The dynamic rectangles, caused by the multiplexing architecture for producers/consumers, must be calculated *every* time the test schedule changes.

Adapted Dynamic Rectangle Packing: The pseudocode for *Adapted_TAM_Schedule_Optimizer* is shown in Algorithm 3. The algorithm takes the core list C , *TIG* and the TAM division as inputs, and it generates the *schedule* for each core and the SOC testing time. It should be noted that in Algorithm 3 we only show the differences with respect to the original *TAM_Schedule_Optimizer* algorithm proposed in [8] (for the sake of self-containment the reader should check [8] for more details on terminology).

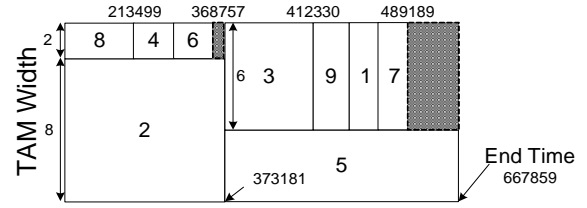
Algorithm 3 - Adapted_TAM_Schedule_Optimizer

INPUT: $C, TIG, W_{prod}, W_{CUT}, W_{cons}$
OUTPUT: $schedule, testing_time$

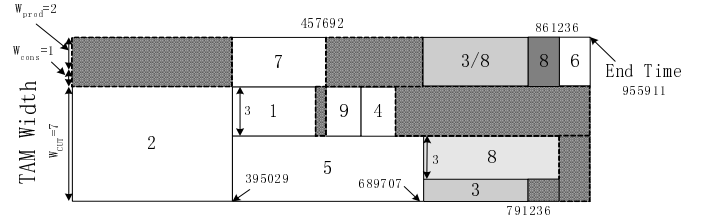
1. Compute collection R_s of rectangles for P1500-compliant core set C_s ;
 2. Initialize(C_s, d, p); /* C_s is the P1500-compliant core set*/
 3. Set $w_{avail} = W_{CUT}$; $current_time = 0$;
 4. **While** $C \neq \emptyset$ {
 5. **if** $w_{avail} > 0$ {
 6. Find unscheduled light-wrapped core set C'_d ;
 7. Compute collection R'_d of dynamic rectangles for C'_d ;
 8. Initialize(C'_d, d, p);
 9. Schedule compatible cores that can be assigned preferred TAM width; (see [8])
 10. Schedule compatible cores that can use the resulting idle TAM wires; (see [8])
 11. Update $L_{prod}, L_{cons}, loadSize$;
 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 $L_{prod}, L_{cons}, loadSize$;
 20. Update $test_time$ for scheduling light-wrapped cores;
 21. Update $this_time$;
 22. **return** $schedule, testing_time$;
-

As described earlier, for light-wrapped cores the test **cannot** be pre-computed and represented as a static rectangle; its testing time (the width of the rectangle) varies with its schedule, hence its rectangle representation is computed dynamically (lines 7 and 15). In [8] the procedure *Initialize* 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 $p = 1.0$; this may result in a different schedule and a slightly longer testing time in some cases when compared to the result in [8]. Whenever a light-wrapped core is scheduled, $L_{prod}, L_{cons}, loadSize$ for currently scheduling light-wrapped cores need to be updated (line 11, 19) and the testing time recalculated (line 12, 20).

If a light-wrapped core has no internal scan chains inside and hence it does not need any TAM lines in the G_{CUT} , we may be able to schedule it even when the available TAM



(a) Test Schedule for m4953 when all the cores are P1500-compliant using [8]



(b) Test schedule when $Core_3, Core_6, Core_7$ and $Core_8$ are light-wrapped using the new method

Figure 6. Test Schedules for m4953.

width $w_{avail} = 0$ in G_{CUT} (lines 15 to 20). This is because only G_{prod} and G_{cons} resources are necessary. 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. To summarize this section, the following example will illustrate (using the m4953 SOC of Figure 3) what is the impact of producers/consumers on the testing time.

Example 5 In Figure 6(a) (given the total TAM width $W = 10$), we present the test schedule obtained for m4953 when all the cores are P1500-compliant. When applying the new *Decide_Wrapper_Type* (algorithm 2) to m4953, if no wrapper design constraints exist, $Core_3, Core_6, Core_7$ and $Core_8$ are selected to be light-wrapped. Using the proposed *TAM_Division_Schedule* (algorithm 1), we obtain $W_{CUT} = 7, W_{prod} = 2$ and $W_{cons} = 1$ as the best possible TAM division and the test schedule is shown in Figure 6(b). We can observe that the testing time increases by approximately 45%, due to the following two 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.

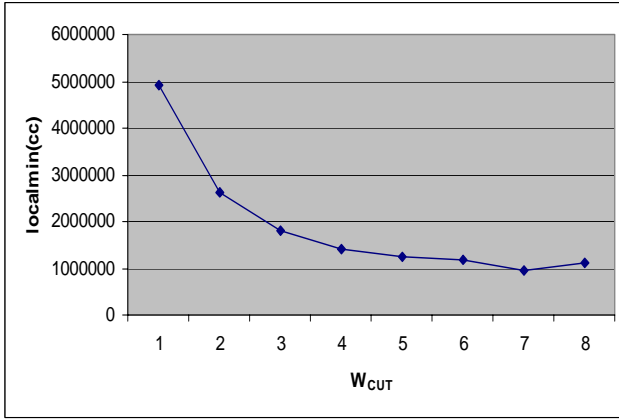


Figure 7. Testing Time Variation with W_{CUT} .

It can be observed in Figure 6(b), that $Core_7$ is scheduled from the same starting time as $Core_1$ even when the available number of TAM lines $G_{CUT} = 0$; this is because $Core_7$ has no internal scan chains and test data can be transferred only using G_{prod} and G_{cons} . Note also that $Core_3$ and $Core_8$ share the producer/consumer TAMs since we use a multiplexing architecture [1]. The testing time variation with W_{CUT} for $m4953$ is shown in Figure 7. For this particular case we deal with a convex function and the first identified local minimum is the only global minimum. If we set weight = 1.1 (see Algorithm 1), the search for the minimum testing time will start from $W_{CUT} = 8$ and stop at $W_{CUT} = 6$. This will prune the search space and hence reduce the computational time while getting the best possible TAM division and test schedule.

4 Experimental Results

To investigate the implication of the proposed solution on the testing time and the number of light-wrapped cores (and hence DFT area and performance improvements) experiments were carried out for the benchmark SOCs provided originally from the ITC02 SOC test benchmarking initiative [11]. Since the functional interconnects are not provided in the benchmark files we have randomly generated them to support the proposed approach, by including the direct connection between cores and functional busses. 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 testing time. We have assumed that the SOCs have $\lceil \frac{N_c}{10} \rceil$ (for $N_c > 5$) busses and each bus has a random number p ($3 \leq p \leq \max(N_c, 8)$) of cores attached to it, where N_c is the total number of cores in the SOC. We have also assumed that every core has a random number q ($1 \leq q \leq 3$) of producers. According to the randomly generated functional interconnect topology, we have assumed that there are no wrapper design constraints and the proposed solution determines the wrapper type of each core, the optimal TAM division and the test schedule.

SOC	N_c	$N_{max,L}$	$N_{min,L}$	$N_{ave,L}$	$\Delta N_l(\%)$
d695	10	6	3	4.22	42.20
g1023	14	7	5	5.66	40.43
p22810	28	14	9	11.56	41.29
p34392	19	10	7	8.32	43.79
p93791	32	15	12	13.46	42.06
t512505	31	15	11	12.66	40.84

Table 1. The Number of Light-Wrapped Cores.

Table 1 shows the number of light-wrapped cores in the 50 random-generated interconnects for several SOCs [11]. N_c is the number of cores, and N_{max} , N_{min} and N_{ave} denote the maximum, minimum and average number of light-wrapped cores. ΔN_l is calculated using the formula $\Delta N_l(\%) = \frac{N_{ave}}{N_c} \times 100$. This result shows that to provide the same test quality for core-based SOCs a high number of cores (approximately 40%) need not to be wrapped with WBR cells. Based on functional interconnect topology there are cases where the maximum number of unwrapped cores can be half of the total number of cores (see column 3 in Table 1). This result, however, does not show the average number of light-wrapped cores when there are core wrapper design constraints, which will, however guarantee that performance specifications are not violated. These problems are investigated in our ongoing research.

Tables 2, 3 and 4 present testing time results when varying the total TAM width W . T_{ave} , T_{max} and T_{min} denote the average, maximum and minimum testing time. The percentage change is calculated using the formula $\Delta T(\%) = \frac{T_{ave} - T}{T} \times 100$, where T is the testing time result obtained using the algorithm described in [8]; it should be noted that since we did not manually select the d and p parameters, the testing time T is slightly increased when compared to the result reported in [8]. As shown in Tables 2, 3 and 4 the average increase in testing time can vary from 43% to 171%. For g1023 the penalty is higher than for p34392 and p93791. This is because, in addition to the reasons analyzed earlier in Example 5, the number of internal scan flip flops in g1023 is comparable to the number of the producers'/consumers' outputs/inputs. Hence a large amount of testing time is necessary to load/unload test stimuli/response, thus leading to a higher number of TAM lines assigned to producer/consumer TAMs, which leads to less TAM lines for CUTs. It can also be observed that the difference between the maximum and minimum testing time 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 testing time will increase significantly. However, this penalty in testing time can be greatly improved simply by wrapping the large cores (that are involved in many test conflicts) with P1500-compliant wrappers.

SOC g1023					
W	[8]	NEW LIGHT-WRAPPED			
	$T(cc)$	$T_{max}(cc)$	$T_{min}(cc)$	$T_{ave}(cc)$	$\Delta T(\%)$
8	66423	309969	102758	180149	+171.21
16	35156	151726	50673	88732	+152.40
24	24018	104247	33352	61157	+154.63
32	19620	80463	28777	48298	+146.17

Table 2. Testing Time Comparison for g1023.

SOC p34392					
W	[8]	NEW LIGHT-WRAPPED			
	$T(cc)$	$T_{max}(cc)$	$T_{min}(cc)$	$T_{ave}(cc)$	$\Delta T(\%)$
16	1075242	2926688	1396603	2149945	+99.95
32	544579	1607899	847033	1206576	+121.56
48	544579	1140900	620895	923081	+69.50
64	544579	1067460	554999	816840	+49.99

Table 3. Testing Time Comparison for p34392.

SOC p93791					
W	[8]	NEW LIGHT-WRAPPED			
	$T(cc)$	$T_{max}(cc)$	$T_{min}(cc)$	$T_{ave}(cc)$	$\Delta T(\%)$
16	1901700	3906006	2266078	2725138	+43.31
32	1053329	2770435	1220977	1523516	+44.64
48	640190	1389335	779275	1031337	+61.10
64	544052	1202553	624275	785284	+44.34

Table 4. Testing Time Comparison for p93791.

5 Conclusion

This paper has described a P1500-compliant [13] modular SOC testing methodology based on *producers, consumers and light-wrapped cores*. Motivated by the escalating DFT area and performance overhead, caused by wrapping all the embedded cores in the system, we have shown how a large number of wrapper boundary register cells can be removed without affecting the test quality. 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 it does not rely on the test or diagnosis data at hand. The lower test concurrency may have an impact on the overall cost of SOC testing, however since we increase only the number of clock cycles required to apply the tests, for a given tester channel capacity, the volume of test data is not changed. Therefore, if the memory management capabilities of the new DFT testers [2] are exploited, such as the reconfigurable memory pool feature, the number of tester buffer reloads, which dominates the overall time a digital chip spends on the tester, will be similar to the case when all the cores in the SOC are wrapped. This makes the proposed solution particularly attractive, since it is capable to decrease the DFT area requirements and to reduce the propagation delays, which can easily satisfy or improve SOC's functional performance.

References

- [1] J. Aerts and E. J. Marinissen. Scan Chain Design for Test Time Reduction in Core-Based ICs. In *Proceedings IEEE International Test Conference (ITC)*, pages 448–457, Washington, DC, Oct. 1998.
- [2] J. Bedsole, R. Raina, A. Crouch, and M. S. Abadir. Very Low Cost Testers: Opportunities and Challenges. *IEEE Design and Test of Computers*, 18(5):60–69, September 2001.
- [3] K. Chakrabarty. Design of System-on-a-Chip Test Access Architectures Using Integer Linear Programming. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 127–134, Montreal, Canada, Apr. 2000.
- [4] P. T. Gonciari, B. M. Al-Hashimi, and N. Nicolici. Useless memory allocation in system-on-a-chip test: Problems and solutions. In *Proc. IEEE VLSI Test Symposium*, pages 423–429, April 2002.
- [5] International SEMATECH. *The International Technology Roadmap for Semiconductors (ITRS): 2001 Edition*. <http://public.itrs.net/Files/2001ITRS/Home.htm>, 2001.
- [6] V. Iyengar, K. Chakrabarty, and E. Marinissen. Test wrapper and test access mechanism co-optimization for system-on-chip. *Journal Electronic Testing: Theory and Applications (JETTA)*, 18:211–228, March 2002.
- [7] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip. In *Proceedings IEEE International Test Conference (ITC)*, pages 1023–1032, Baltimore, MD, Oct. 2001.
- [8] V. Iyengar, K. Chakrabarty, and E. Marinissen. On using rectangle packing for SOC wrapper/TAM co-optimization. In *Proc. IEEE VLSI Test Symposium*, pages 253–258, 2002.
- [9] E. Larsson and Z. Peng. An Integrated System-on-Chip Test Framework. In *Proceedings Design, Automation, and Test in Europe (DATE)*, pages 138–144, Munich, Germany, Mar. 2001.
- [10] E. J. Marinissen et al. A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores. In *Proceedings IEEE International Test Conference (ITC)*, pages 284–293, Washington, DC, Oct. 1998.
- [11] E. J. Marinissen, V. Iyengar, and K. Chakrabarty. ITC'02 SOC Test Benchmarks Web Site. <http://www.extra.research.philips.com/itc02socbench/>.
- [12] E. J. Marinissen, V. Iyengar, and K. Chakrabarty. A Set of Benchmarks for Modular Testing of SOCs. In *Proceedings IEEE International Test Conference (ITC)*, Baltimore, MD, Oct. 2002.
- [13] P1500 SECT Task Forces. IEEE P1500 Web Site. <http://grouper.ieee.org/groups/1500/>.
- [14] N. Touba and B. Pouya. Using Partial Isolation Rings to Test Core-Based Designs. *IEEE Design & Test of Computers*, 14(4):52–59, Dec. 1997.
- [15] Y. Zorian, S. Dey, and M. Rodgers. Test of future system-on-chips. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 392–398, San Jose, CA, Nov. 2000.