Wrapper/TAM Co-Optimization, Constraint-Driven Test Scheduling, and Tester Data Volume Reduction for SOCs*

Vikram Iyengar[1] Krishnendu Chakrabarty[1] Erik Jan Marinissen[2]

[1]Electrical & Computer Engineering, Duke University, Durham, NC 27708, USA
vik@ee.duke.edu krish@ee.duke.edu
[2]Philips Research Laboratories, 5656 AA Eindhoven, The Netherlands
erik.jan.marinissen@philips.com

ABSTRACT

This paper describes an integrated framework for plug-and-play SOC test automation. This framework is based on a new approach for wrapper/TAM co-optimization based on rectangle packing. We first tailor TAM widths to each core's test data needs. We then use rectangle packing to develop an integrated scheduling algorithm that incorporates precedence and power constraints in the test schedule, while allowing the SOC integrator to designate a group of tests as preemptable. Finally, we study the relationship between TAM width and tester data volume to identify an effective TAM width for the SOC. We present experimental results for non-preemptive, preemptive, and power-constrained test scheduling, as well as for effective TAM width identification for an academic benchmark SOC and three industrial SOCs.

Categories and Subject Descriptors

B.7.3 [**Integrated circuits**]: Reliability and testing

General Terms

Algorithms,design,experimentation,reliability,verification

1. INTRODUCTION

Test automation is widely recognized as an important part of the overall system-on-chip (SOC) design and integration cycle. In order to reduce design cycle time and provide greater functionality, a large number of intellectual property (IP) cores are stitched into an SOC. To facilitate plug-and-play test, an embedded core must be isolated from surrounding logic, and test access must be provided from the I/O pins of the SOC. Test wrappers are used to isolate the core for modular test application, while test access mechanisms (TAMs) transport test patterns and test responses between SOCs pins and core I/Os [18]. In addition, core tests must be scheduled such that precedence and power constraints are met and conflicts in the TAM are avoided.

To reduce cost and ensure quality, testing must make effective use of SOC test resources [6]. The rapidly increasing size of SOCs has spurred an enormous growth in test resource usage, leading to complex test hardware, long test application times, and large test data sets. An integrated framework for SOC test automation that increases the utilization of test resources is therefore necessary to increase production capacities and reduce test cost. This paper describes the design of such a framework that integrates the following three design processes into the test automation flow.

1. Wrapper/TAM co-optimization. Test wrapper design and TAM optimization are of critical importance during system integration since they directly impact hardware overhead, testing time and tester data volume on the tester. The new approach for TAM optimization is based on a generalized version of rectangle packing [8, 14].

2. Constraint-driven preemptive test scheduling. The objective of the proposed scheduling algorithm is to minimize testing time, while addressing the following issues: (a) resource conflicts between cores arising from the use of shared TAMs and on-chip BIST engines, (b) precedence constraints among tests, and (c) power dissipation constraints. Testing time is decreased further through the selective use of test preemption.

3. Tester data volume reduction. The third component is used to identify an SOC TAM width that reduces testing time as well as tester data volume. Test data for large SOCs now require several Gigabits of tester memory, which is a significant contributor to test cost [3, 6]. The impact of TAM design and test schedules on tester memory requirement has not been directly studied before. We show here that TAM widths that minimize testing time do not always lead to minimum tester data volume. We investigate the correlation of TAM width to the tester data volume, and determine TAM widths that minimize a cost function involving both the testing time and the amount of tester data. This allows the system integrator to trade-off testing time with data volume.

2. PRIOR WORK

Most prior research in test access for SOCs has either studied wrapper design and TAM optimization as independent problems, or not addressed the issue of sizing TAMs to minimize SOC testing time [1, 4, 18]. Alternative approaches that combine TAM design with test scheduling [9, 16, 19] do not address the problem of wrapper design and its relationship to TAM optimization.

The first integrated wrapper/TAM co-optimization methodologies were proposed in [12, 13]. A drawback of [12] is that the problem of wrapper/TAM co-optimization as formulated is intrinsically intractable; the compute time for the exact method of [12] increases exponentially with the number of TAMs. Efficient heuristics to reduce CPU time for wrapper/TAM design were presented in [13]. However, the approaches in [12, 13] are limited to test access architectures based on "fixed-width" TAMs. In fixed-width TAMs, the total TAM width is explicitly partitioned among a finite number of TAMs, and each core is assigned to exactly one TAM.

*This research was supported in part by the National Science Foundation under grant number CCR-9875324 and by an IBM Graduate Fellowship.

DAC 2002, June 10-14, 2002, New Orleans, Louisiana, USA.

Such architectures are inflexible and they generally lead to inefficient usage of TAM wires [14]

Several techniques for test scheduling of SOCs have been proposed [5, 7, 11, 15, 16, 18]. Methods to incorporate precedence and power constraints in a preemptive test schedule were presented in [11]. These methods assume that a pre-designed TAM for the SOC is provided. We address the design of an integrated framework for SOC test automation where TAM optimization and test scheduling are performed in conjunction. An integrated TAM design and test scheduling method based on flexible-width TAMs was presented in [14]. However, precedence and power constraints in the test schedule and test preemption were not considered.

Tester data volume reduction methods are either based on built-in self test (BIST) [2] or test data compression [6]. While both methods have been shown to be useful for test data volume reduction, there has not been a concerted effort to examine their use coupled with TAM design and test scheduling in the form of an integrated framework for test automation.

In this paper, we present a novel approach to integrated wrapper/TAM co-optimization and test scheduling based on a generalized version of rectangle packing [8]. A rectangle representation of core tests, similar to the one presented in [9], is used in our co-optimization and scheduling framework. We partition the TAM width among the set of of cores being tested during each interval of time. This partition is varied with time to match the TAM width to each core's test data needs. TAMs can fork and merge between cores to improve TAM wire utilization. Precedence constraints among tests can be embedded into the schedule by the system integrator, and the maximum power constraint that must not be exceeded during test can be easily incorporated. Moreover, the system integrator can identify certain tests as preemptable, as well as specify limits on the number of preemptions allowed for each test. Finally, the relationship between TAM width and tester data volume is investigated to identify an effective choice of total TAM width for the SOC.

3. TAM DESIGN AND TEST SCHEDULING

The general integrated wrapper/TAM co-optimization and test scheduling problem that we address in this paper is as follows. We are given the total SOC TAM width W, and the test set parameters for each core, i.e., the number of primary I/Os, test patterns, scan chains, and the scan chain lengths. Unlike in [1], we assume that the lengths of scan chains are fixed. The goal is to determine the TAM width and a wrapper design for each core, and a test schedule that minimizes the testing time for the SOC such that the following constraints are satisfied.

1. The total number of TAM wires utilized at any moment does not exceed W;

2. Precedence and concurrency constraints are met;

3. The maximum power dissipation value is not exceeded;

4. Selective preemption of tests is allowed.

Additionally, we would like to determine a value of W for the SOC to trade-off testing time with tester data volume.

We formulate this problem as a progression of three problems of increasing complexity that lead up to the general problem. These three problems are as follows:

Problem 1. Wrapper/TAM co-optimization and test scheduling.

Problem 2. Wrapper/TAM co-optimization and test scheduling with selective preemption, and precedence and power constraints.

Problem 3. Wrapper/TAM co-optimization, test scheduling with selective preemption, and precedence and power constraints, and identification of a TAM width to trade-off testing time with tester data volume.

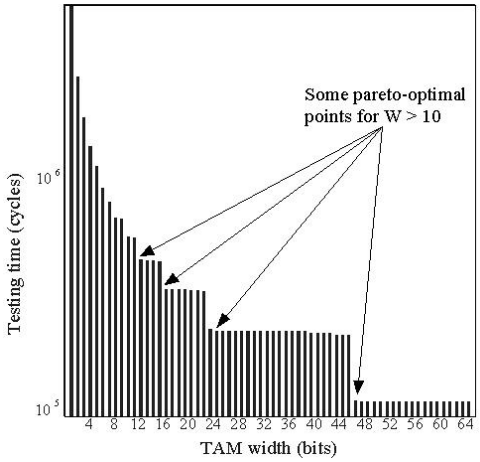

Figure 1: Relationship between testing time and TAM width for Core 6 in Philips SOC p93791.

In this section, we address *Problem 1*, and show how wrapper/TAM co-optimization can be integrated with test scheduling. In Section 4, we show how this problem is generalized to include precedence, preemption and power-constraints–*Problem 2*. Finally, in Section 5, we study *Problem 3*, i.e., we identify TAM widths that provide a trade-off between test time and tester data volume.

Problem 1: Given W and the test set parameters for each core, determine the TAM width and a wrapper design for each core, and a test schedule for the SOC that minimizes the total testing time, such that the total number of TAM wires utilized at any moment does not exceed W. □

We solved the problem of wrapper design for cores in [12] using the *Design_wrapper* algorithm based on the Best Fit Decreasing (BFD) heuristic. In order to solve the problem of assigning TAM width to cores and scheduling tests, we represent core tests by rectangles. The *Design_wrapper* algorithm is used to construct a set of rectangles representing the different testing times for each value of TAM width for a core, such that the height of the rectangle corresponds to the TAM width and the width of the rectangle represents the core test application time. The use of rectangles for core test representation during test scheduling has previously been studied in [7, 9, 10, 16]. A bin packing approach based on rectangle representation was used in [10] for test scheduling.

For a given core, the testing time decreases only at Pareto-optimal points when the TAM width exceeds core-specific thresholds [12]. Pareto-optimal points are formally defined in [14]. For example, in Figure 1, a TAM width of 46 results in a testing time of 115850 cycles, while all TAM widths from 47 up to 64 result in the same testing time of 114317 cycles. Hence 47 is a Pareto-optimal TAM width, and rectangles of height between 48 and 64 can be ignored. The TAM width assigned to cores is thus always the minimal value required to achieve a specific testing time. The extra TAM wires can be used for other cores in the SOC, thereby yielding a more efficient TAM design.

We now formulate *Problem 1* as a generalized version of the rectangle packing problem [8]. Consider an SOC having N cores, and let $\mathbf{R}_i$ be the set of rectangles for core i, $1 \leq i \leq N$. Problem $\mathcal{P}_{\mathrm{GRP}}1$ (generalized rectangle packing) is stated as follows.

$\mathcal{P}_{\mathrm{GRP}}1$: Select one rectangle $R_{ij} \in \mathbf{R}_i$ from each set $\mathbf{R}_i$, $1 \leq i \leq N$, and pack the selected rectangles into a bin of fixed height and unbounded width, such that no two rectangles overlap, and the width to which the bin is filled is minimized. Furthermore, during packing, each rectangle selected is allowed to be split vertically into several non-adjacent pieces, each having the same co-ordinates

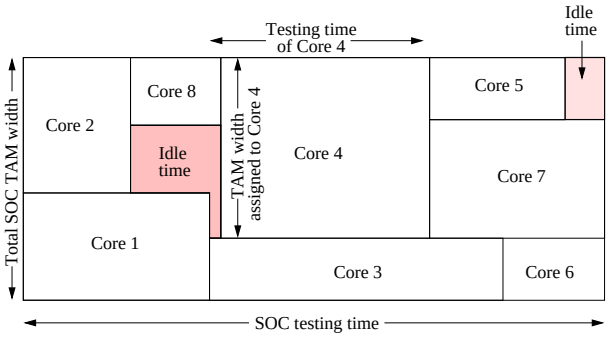

Figure 2: **Example test schedule using rectangle packing.**

on the horizontal axis. □

In Problem $\mathcal{P}_{\rm GRP}1$, during packing, the rectangle selected for a core can be vertically split into several non-adjacent rectangles having the same width. This is because it is possible to assign a group of non-contiguous TAM wires to a single core, using fork-and-merge of TAM wires.

Problem $\mathcal{P}_{\rm GRP}1$ relates to *Problem 1* as follows; see Figure 2. The height of the rectangle for each core corresponds to the TAM width assigned to the core, while the rectangle width corresponds to the testing time. The height of the bin corresponds to the total SOC TAM width, and the width to the which the bin is filled corresponds to the SOC testing time. The unfilled area of the bin corresponds to the idle time on TAM wires during test. Furthermore, the distance between the left edge of each rectangle and the left edge of the bin corresponds to the beginning time of each core test. Thus a one-to-one correspondence exists between the packed bin and the final test schedule.

Problem $\mathcal{P}_{\rm GRP}1$ can be shown to be $\mathcal{NP}$-hard by a restriction argument. A special case of $\mathcal{P}_{\rm GRP}1$ in which the cardinality of each set $\mathbf{R}_i$, $1 \leq i \leq N$ equals one directly corresponds to the rectangle packing problem in [8]. Since the rectangle packing problem was shown to be $\mathcal{NP}$-hard in [8] (by restriction to Bin Packing), $\mathcal{P}_{\rm GRP}1$ is also $\mathcal{NP}$-hard.

4. CONSTRAINED TEST SCHEDULING

In this section, we first detail *Problem 2* (integrated TAM design and constraint-driven test scheduling), and then formulate Problem $\mathcal{P}_{\rm GRP}2$, a generalized version of $\mathcal{P}_{\rm GRP}1$ that is equivalent to *Problem 2*.

Problem 2: Solve *Problem 1*, such that:

1. Precedence and concurrency constraints are met;

2. The maximum power dissipation value P_{max} is not exceeded;

3. Selective preemption of tests is allowed.

Precedence constraints reduce test cost and increase test efficiency. For example, "abort at first fail" strategies order tests such that either the cores most likely to fail are tested first, or tests that detect more faults are applied first [15]. Furthermore, memories are often tested and diagnosed earlier so that they can be used later for system test. Concurrency constraints seek to avoid conflicts in the test hardware. For example, a hierarchical parent core cannot be tested at the same time as the child cores lying within it. This is because the wrappers of the child cores must be in Extest mode while the parent core is being tested in Intest mode. Finally, power constraints must be incorporated in the schedule to ensure that the power rating of the SOC is not exceeded during test.

Problem 2 can be expressed in terms of rectangle packing as follows. Consider an SOC having N cores, and:

1. Let $\mathbf{R}_i$ be the set of rectangles for core i, $1 \leq i \leq N$;

2. Let precedence constraints between tests be defined, e.g., $i < j$ (test i must complete before test j is begun);

3. Let concurrency constraints between tests be defined, e.g., $i <> j$ (test i must not be applied at the same time as test j);

4. Let the test for Core i have a power dissipation of P_i;

5. Let each core be assigned a maximum number of allowed preemptions.

$\mathcal{P}_{\rm GRP}2$: Solve Problem $\mathcal{P}_{\rm GRP}1$, while allowing each rectangle to be horizontally split into several smaller, non-adjacent rectangles, each having the same height (the rectangles are split along the time axis). The number of such partitions for a rectangle must not exceed the maximum number of preemptions allowed for the test. For each precedence constraint $i < j$, the rectangle for Core i can be packed only after all partitions of the rectangle for Core j are packed. Furthermore, the rectangle representing test i must not overlap (in time) the rectangle for test j, if there is a specified concurrency constraint $i <> j$. Finally, at any moment of time the sum of the P_i values for the rectangles selected must not exceed the maximum specified value P_{max}. □

Problem $\mathcal{P}_{\rm GRP}2$ is a generalized version of $\mathcal{P}_{\rm GRP}1$, and can therefore be shown to also be $\mathcal{NP}$-hard by a restriction argument.

Next, we describe our solution to Problem $\mathcal{P}_{\rm GRP}2$. Our algorithm first identifies a "preferred TAM width" ($width_p(i)$) for each Core i such that the core's testing time is within a small percentage of its testing time at a maximum allowable TAM width W_{max}. (In this paper, W_{max} is chosen to be 64.) A group of tests is then selected to be scheduled, such that precedence and power constraints are met, and BIST–scan test conflicts are not created. If the number of available TAM wires is insufficient to add a new test to the group, the resulting idle time is filled using several heuristics that seek to insert tests to minimize the idle time. If idle time is inevitable, the algorithm waits until the first currently-running test completes, and then repeats the scheduling process for the remaining tests. Tests may be preempted and resumed again such that the number of preemptions does not exceed a designer-specified value for each test. Next, we explain the rationale behind the heuristic decision-making in our algorithm, and show how these decisions minimize system testing time.

Data structure. The data structure in which we store the TAM width and testing time values for the cores of the SOC is presented in Figure 3. This data structure is updated with the assigned TAM width, begin and end times, and preemption count for each core as the test schedule is developed.

Preferred TAM widths. The pseudocode for our TAM optimization and test scheduling algorithm is presented in Figure 4. The inputs to our algorithm are the set **C** of cores, total TAM width W, set **PC** of precedence constraints, set **CC** of concurrency constraints, power constraint P_{max}, and user-input parameters p and d (explained later). In Procedure *Initialize* (Line 1), we calculate the collection **R** of Pareto-optimal rectangles as described in Section 3, and the preferred TAM width values for each core from the input percent value p. Recall that the core testing time varies with TAM width w as a staircase function that drops rapidly at first for small

Data structure *Schedule*
1 $width_p(i)$ /* preferred TAM width for Core i */
2 $width(i)$ /* TAM width assigned to Core i */
3 $first\_begin(i)$ /* first begin time of Core i */
4 $end(i)$ /* end time of Core i */
5 $scheduled\_times(i)$ /* times during which Core i is scheduled */
6 $time\_left(i)$ /* testing time remaining for Core i */
7 $begun(i)$ /* boolean indicates Core i has begun at least once */
8 $scheduled(i)$ /* boolean indicates Core i is scheduled */
9 $complete(i)$ /* boolean indicates test for Core i has completed */
10 $preempts(i)$ /* number of times Core i has been preempted */
11 $max\_preempts(i)$ /* max. number of preemptions allowed */

Figure 3: **Data structure for the test schedule.**

Procedure *TAM_schedule_optimizer*(**C**, W, **PC**, **CC**, P_{max}, d, p)
1 *Initialize*(**C**, d, p);
2 Set $w\_avail = W$; $current\_time = 0$;
3 **While C** $\neq \emptyset$
4 **If** $w\_avail > 0$
5 **If** a Core $i \in$ **C** can be found, such that
$begin(i) < current\_time$ AND $scheduled(i) = 0$ AND
$preempts(i) = max\_preempts(i)$
6 *Assign*(i, $width(i)$, 0);
7 **If** a Core $i \in$ **C** can be found, such that
$begun(i) = 1$ AND $width(i) \leq w\_avail$ AND
$time\_left(i)$ is maximum AND (*Conflict*(i, **PC**, **CC**, P_{max})=0)
8 **If** $end(i) < current\_time$
9 *Assign*(i, $width(i)$, 1);
10 **Else** *Assign*(i, $width(i)$, 0);
11 **If** a Core $i \in$ **C** can be found, such that
$begun(i) = 0$ AND $width_p(i) \leq w\_avail$ AND
$time\_left(i)$ is maximum AND (*Conflict*(i, **PC**, **CC**, P_{max})=0)
12 *Assign*(i, $width_p(i)$, 0);
13 **If** a Core $i \in$ **C** can be found, such that
$begun(i) = 0$ AND $width_p(i) \leq w\_avail + 3$ AND
$width_p(i)$ is minimum AND (*Conflict*(i, **PC**, **CC**, P_{max})=0)
14 *Assign*(i, w_avail, 0);
15 **If** a Core $i \in$ **C** can be found, such that
$first\_begin(i) = current\_time$ AND
$T_i(width(i)) - T_i(width(i) + w\_avail)$ is maximum;
16 *Assign*(i, $width(i) + w\_avail$, 0);
17 **Else** *Update*(**C**);
18 **Return** *Schedule*;

Figure 4: **Algorithm for solving $\mathcal{P}_{\mathrm{GRP}}2$.**

Procedure *Initialize*(**C**, d, p)
1 Compute collection **R** of rectangles using *Design_wrapper*;
2 **For** each Core $i \in$ **C**
3 Calculate $T_{ip} = T_i(W_{max}) + \frac{p}{100} \times (T_i(1) - T_i(W_{max}))$;
4 Set $width_p(i) = w$, such that $T_i(w) - T_{ip}$ is minimum;
5 Calculate highest Pareto-optimal width w_h;
6 **If** $w_h - width_p(i) \leq d$ **then** $width_p(i) = w_h$;

Figure 5: **Preferred widths initialization subroutine.**

values of w and less rapidly after that. For example, for Core 6 in p93791 (Figure 1), at $w = 10$ the testing time reaches within 10% of its value at $w = 64$, and at $w = 15$, the testing time is within 5% of its value at $w = 64$. Hence, instead of attempting to assign the highest Pareto-optimal width 47 to this core, a considerable savings in system TAM width can be realized by assigning a pre-calculated value, such that the testing time of the core reaches within a small percent value p of its testing time at $w = W_{max}$. The value of p is usually between 1 and 10.

Initializing subroutine. In subroutine *Initialize* (Figure 5), we initialize $width_p(i)$ to the preferred TAM width T_{ip}. ($T_i(w)$ denotes the testing time of Core i, with a TAM width of w.) In Lines 5 and 6 of *Initialize*, we allow $width_p(i)$ to be set to the highest Pareto-optimal width w_h if the difference between $width_p(i)$ and w_h is less than the input difference value d. This heuristic aids significantly in minimizing system testing time, when it is beneficial to assign a few ($\leq d$) extra TAM wires to a bottleneck core. For example, when using $p = 2$ for example SOC p34392 (presented in Section 6), we noticed that Core 18 was assigned 9 bits, leading to $T_{18}(9) = 622163$ cycles. The testing time for the SOC was also found to be 622163 cycles, from which we noted that Core 18 is a bottleneck core. Furthermore, Core 18's highest Pareto-optimal width is 10 bits, at which the testing time for Core 18 reaches its minimum value of 544579 cycles. Hence, providing an extra TAM wire to Core 18 reduced its testing time as well as the overall SOC testing time to $T_{18}(10) = 544579$ cycles. Thus the minimum testing time for SOC p34392 could be achieved using the heuristic in Lines 5 and 6 of *Initialize* with $d = 1$.

TAM width assignment and test scheduling. Line 2 of Figure 4 initializes the main rectangle packing loop. While executing the main **While** loop (Line 3), if there are w_avail TAM wires available for assignment, cores are assigned to the TAM using a three-priority selection mechanism:

Priority 1: Line 5 searches for a core i whose test has already been preempted $max\_preempts(i)$ times, but has not completed. If such a core is found it is scheduled using the *Assign* subroutine (Figure 6). After *Assign* completes, control is returned to Line 4 of Figure 4, and the process of selecting a core begins again. Thus only if no core is found by *Priority 1*, does*Priority 2* come into play.

Priority 2: If a core is found, whose test has begun earlier, whose assigned TAM width is less than w_avail, and whose remaining testing time is largest among all such cores, then it is scheduled using *Assign* in Lines 7 to 10.

Priority 3: If a core is found, whose test has not begun earlier, whose preferred TAM width is less than w_avail, and whose remaining testing time is largest among all such cores, then it is scheduled using *Assign* in Lines 11 to 12.

Priority 1 is motivated by the need to complete the test for cores that cannot be preempted further, while *Priorities 2* and *3* seek to assign the preferred TAM width to each core.

Precedence, concurrency and power constraints. During selection of a core to be scheduled in Lines 7, 11, and 13 of Figure 4, the *Conflict* subroutine (Figure 7) is invoked to ensure that (i) precedence conflicts, (ii) concurrency conflicts, and (iii) power constraint conflicts are avoided.

Rectangle insertion in idle time. If there is no core found in Lines 5 to 12, rather than let the w_avail TAM wires remain idle, *TAM_schedule_optimizer* attempts to insert the rectangle for some unscheduled core into the available time. In Line 13, we find a core that has not been scheduled and whose preferred TAM width is within 3 bits of w_avail. This core is then scheduled using *Assign*. The 3-bit limit was found to be the most useful after extensive experimentation. This is because, in general, as long as the width of the rectangle chosen is within 3 bits of the preferred width, the gains to testing time reduction are greatest. However, if a value other than 3 is found to be more useful for another group of SOCs, this can be easily entered into our program by the system integrator during test automation.

Increasing TAM widths to fill idle time. If no rectangle is available to fill in the idle time, then the heuristic in Lines 15 to 16 is used to determine which of the cores currently scheduled to begin at $current\_time$ will benefit the most, in terms of testing time decrease, from an extra w_avail TAM wires. If such a core can

Procedure *Assign*(i, $width$, $preempt$)
1 Let i be the core to be scheduled;
2 Set $width(i) = width$; $w\_avail = w\_avail - width$;
3 Set $scheduled(i) = 1$; $preempts(i) = preempts(i) + preempt$;
/*s_i, s_o are the longest wrapper scan-in and scan-out lengths [12]*/
5 **If** $preempt = 1$, Set $time\_left(i) = time\_left(i) + \min\{si, so\}$;
6 **If** $begun(i) = 0$
7 Set $begun(i) = 1$; $first\_begin(i) = current\_time$;
9 Set $time\_left(i) = T_i(width(i))$;
10 Set $end(i) = current\_time + time\_left(i)$;
11 **Return** to Line 4 of *TAM_schedule_optimizer*;

Figure 6: **The core assign algorithm.**

Procedure *Conflict*(i, **PC**, **CC**, P_{max})
1 Let i be the core to be scheduled;
2 **For** all precedence constraints $j < i$
3 **If** $complete(j) = 0$, **Return** 1;
4 **For** all concurrency constraints $i <> j$
5 **If** $scheduled(j) = 1$, **Return** 1;
6 Let $P = 0$;
7 **For** all cores $j \neq i$
8 **If** $scheduled(j) = 1$, $P = P + P_j$;
9 **If** $P > P_{max}$, **Return** 1;
10 **For** all cores $j \neq i$
11 **If** there is a BIST–scan test conflict between Cores i and j, **Return** 1;
12 **Return** 0;

Figure 7: **Precedence, concurrency and power constraints.**

Procedure *Update*(**C**)
1 Find Core i such that $time\_left(i)$ is minimum;
2 Set $next\_time = current\_time + time\_left(i)$;
3 **For** all Cores i such that $scheduled(i) = 1$
4 Set $scheduled(i) = 0$; $end(i) = next\_time$;
5 Set $time\_left(i) = time\_left(i) - (next\_time - current\_time)$;
6 **If** $time\_left(i) = 0$
7 Set $complete(i) = 1$; **C** = **C** $- i$;
8 Update $scheduled\_times(i)$;
9 Set $current\_time = next\_time$; $w\_avail = W$;

Figure 8: **Test schedule update algorithm.**

be found, then its currently-assigned $width(i)$ is increased to the highest Pareto-optimal width less than $width(i) + w\_avail$.

Finally, if the heuristics in Lines 4 to 16 fail to find a core to assign, the value of w_avail is set to 0 and the loop beginning at Line 4 is repeated. When w_avail is found to be 0 in Line 4, the execution proceeds to Line 17, where the process of updating $current\_time$ and w_avail is begun. This is presented in subroutine *Update* in Figure 8. Once $current\_time$ is incremented, the widths assigned to packed rectangles (or parts of rectangles) are fixed and cannot be changed later on in the schedule. The resulting test schedule is output in Line 18.

Test preemption. Tests can be selectively preempted each time *Update* is executed. At each execution of *Update*, the value of w_avail is reset to W, and all the incomplete tests contend for the available TAM width in Lines 4 to 16 of *TAM_schedule_optimizer*. If the maximum limit on preemptions for a certain Core i is reached in Line 5, then Core i is continuously scheduled until it completes. On the other hand if the limit on preemptions is not reached, and Core i is preempted (Line 9), then $time\_left(i)$ is incremented by $\min\{s_i, s_o\}$ in Line 5 of *Assign*. Here, s_i (s_o) is the length of the longest wrapper scan in (out) chain [18]. This increment in testing time is necessary because each time a preemption occurs, an extra scan in or scan out must be performed.

5. TESTER DATA VOLUME REDUCTION

In this section, we present the third part of our SOC test automation framework – identification of a value of W that minimizes a weighted cost function involving both the testing time and the tester data volume.

We first motivate the need for identifying such a TAM width. The cost of testing SOCs is closely related to the testing time and the volume of test data. While the time required to apply digital patterns to the SOC is a relatively small fraction of the total test time, the time required to transfer several Gigabytes of data from a workstation to the tester memory is significant if performed frequently [3]. Techniques to ensure that the test data required per pin is contained to a single tester buffer, are therefore vital. The motivation for trading-off TAM width with testing time and data volume lies in multisite testing, in which several ICs are tested in parallel by a single tester. Reduced TAM widths that contain test data per pin to buffer sizes will allow a larger number of ICs to be tested concurrently, thereby decreasing testing time for the entire production batch. Reducing TAM widths also leads to lower routing complexity. It is therefore important to develop techniques that can identify a low number of TAM wires, and to trade-off testing time and data volume.

We begin by plotting the variation of testing time $\mathcal{T}$ with W. This is shown for Philips SOC p22810 in Figure 9(a). (Results for the remaining example SOCs are shown in Table 2 in Section 6.) Note the progressive decrease in $\mathcal{T}$ at Pareto-optimal points. Next, we plot the variation of tester data volume $\mathcal{M}$ with W (Figure 9(b)). $\mathcal{M}$ varies as a non-monotonic function in W, achieving local minima at the Pareto-optimal W values of the $\mathcal{T}$ curve of Figure 9(a).

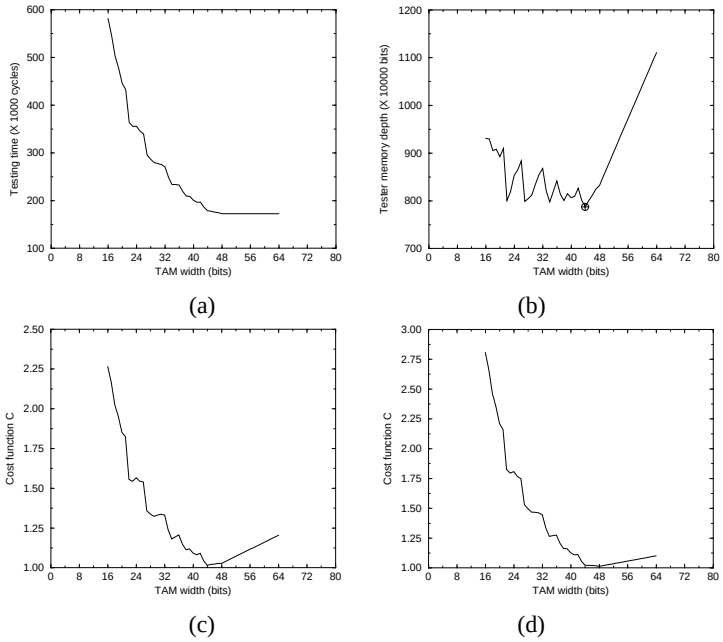

Figure 9: **Relationship between (a) $\mathcal{T}$ and W, and (b) $\mathcal{M}$ and W for SOC p22810; The normalized cost function $\mathcal{C}$ for (c) $\alpha = 0.5$, and (d) $\alpha = 0.75$ for SOC p22810.**

The global minima (marked in Figure 9(b)) is achieved at $W = 44$. However, $W = 44$ does not provide the lowest testing time for the SOC. $\mathcal{T}$ can be decreased by increasing W from 44 to 48 (at which point there is an increase in $\mathcal{M}$). Therefore, by varying W the system integrator can trade-off testing time with tester data volume.

We incorporate this feature in our framework by defining the normalized cost function $\mathcal{C} = \alpha\frac{\mathcal{T}}{\mathcal{T}_{min}} + (1-\alpha)\frac{\mathcal{M}}{\mathcal{M}_{min}}$, where $\mathcal{T}_{min}$ ($\mathcal{M}_{min}$) is the minimum value of $\mathcal{T}$ ($\mathcal{M}$), and $0 \leq \alpha \leq 0$ is a user-input parameter to control the trade-off. As α is varied from 0 to 1, the shape of the $\mathcal{C}$-curve changes from the $\mathcal{M}$-curve to the $\mathcal{T}$-curve. The $\mathcal{C}$-curve is "U" shaped in general having a single minima, illustrated for $\alpha = 0.5$ in Figure 9(c), and for $\alpha = 0.75$ in Figure 9(d). These values of W that minimize $\mathcal{C}$ for various values of α provide the system integrator with effective choices of TAM width for tester data volume reduction. Note that the choice of W will also be affected by the core type and testing method used, e.g., for a core tested only using BIST, the TAM may be very narrow to transport only the BIST control signals. In this paper, we consider only tests that use the TAM for transporting the entire test set.

6. EXPERIMENTAL RESULTS

In this section, we present experimental results for four example SOCs: d695 (an academic SOC from Duke University), p22810, p34392, and p93791 (industrial SOCs from Philips). These four SOCs are part of the ITC 2002 SOC test benchmarks initiative [17]. The experimental results were obtained using a 333 Mhz Sun Ultra 10 with 256 MB memory.

Table 1 presents results on integrated TAM design and test scheduling. We considered all possible integer values of the parameters p and d in the range $1 \leq p \leq 10$, $0 \leq d \leq 4$, and tabulated the best results. A lower bound on the testing time for an SOC can be expressed as $\max\left\{\max_i T_i(W_{\max}), \left\lceil\left(\sum_{i=1}^{N} T_i(1)\right)/W\right\rceil\right\}$. Lower bound values on the testing time for each SOC for several values of W are presented in Table 1. We compare the testing times obtained using non-preemptive and preemptive scheduling. For preemptive testing, the value of $max\_preempts(i)$ was

Table 1: **Wrapper/TAM co-optimization and test scheduling.**

SOC	W	Testing time (cycles) Lower bound	Non-preemptive	Pre-emptive	Preemptive + power-constrained
d695	16	41232	43410	43423	47574
	32	20616	22229	21757	29039
	48	13744	15698	15499	28441
	64	10308	11285	11354	20004
p22810	16	421473	466383	459951	527573
	32	210737	243779	243978	277151
	48	140491	164420	162554	213845
	64	105369	140222	134732	176076
p34392	16	936882	1071043	1082065	1180187
	24	624588	728986	702322	1075971
	28	544579	617018	615126	1075242
	32	544579	544579	544579	1075242
p93791	16	1749388	1860752	1860752	1966092
	32	874694	929311	929311	1247221
	48	583130	637717	643605	656214
	64	437347	503661	492095	631840

set to 2 for the larger cores. Preemptive scheduling obtains lower or equal testing times in most cases. However, in a few cases, non-preemptive schedules are shorter. This is because each preemption adds $\min\{s_i, s_o\}$ cycles to the length of a test, which can increase the overall SOC testing time for SOCs having a large number of short tests that are preempted several times. A careful investigation of the effects of preemption and the use of the $max\_preempts(i)$ parameter considering test lengths is therefore warranted. For p34392, we present testing time results only for $W \leq 32$, since at $W = 32$, we achieve the minimum testing time for p34392, 544579 cycles.

In Table 1, we also present the testing times obtained with power-constraints. We assigned a hypothetical power value P_i to the test for each Core i based on the number of test data bits per test pattern for Core i. The value of P_{max} was set to $\max_i\{P_i\}$ for test scheduling. The increases in testing times for power-constrained scheduling reflect this concurrency constraint. The CPU time of our new algorithm is less than 5 seconds in all cases; this is several orders of magnitude lower than the CPU times required by the method in [12].

Next, Table 2 presents results on effective TAM widths for tester data volume reduction. We present the minimum values of $\mathcal{T}$ and $\mathcal{M}$ obtained for the four SOCs, as well as the values of W at which they occur. We then present the values of $\mathcal{C}_{min}$ and the resulting effective TAM widths W_e obtained for several different values of α. Finally, the corresponding values of $\mathcal{T}$ and $\mathcal{M}$ obtained for these values of W_e are shown. It is clear from Table 2 that the system integrator can trade-off testing time with tester data volume by varying α between 0 and 1. For example, for SOC p22810, the minimum value of $\mathcal{T}$ (14022 cycles) is achieved at $W = 63$. However, the minimum values of $\mathcal{M}$ (7202214 bits) is actually achieved at $W = 18$. By setting $\alpha = 0.3$, the system integrator obtains a TAM width of 48 bits, at which $\mathcal{T} = 164420$ cycles and $\mathcal{M} = 7892160$ bits.

Table 2: **TAM widths for tester data volume reduction.**

$\mathcal{T}_{min}$ (cycles)	${}^1W_1$ (bits)	$\mathcal{M}_{min}$ (bits)	${}^2W_2$ (bits)	α	$\mathcal{C}_{min}$	${}^3W_e$ (bits)	$\mathcal{T}$ at W_e (cycles)	$\mathcal{M}$ at W_e (bits)
d695								
11285	63	675554	22	0.1	1.031	60	11612	696720
				0.3	1.031	60	11612	696720
				0.5	1.026	63	11285	710955
p22810								
$\mathcal{T}_{min}$	${}^1W_1$	$\mathcal{M}_{min}$	${}^2W_2$	α	$\mathcal{C}_{min}$	${}^3W_e$	$\mathcal{T}$ at W_e	$\mathcal{M}$ at W_e
140222	63	7377480	44	0.01	1.018	44	167670	7377480
				0.3	1.103	48	164420	7892160
				0.5	1.093	55	148005	8140275
p34392								
$\mathcal{T}_{min}$	${}^1W_1$	$\mathcal{M}_{min}$	${}^2W_2$	α	$\mathcal{C}_{min}$	${}^3W_e$	$\mathcal{T}$ at W_e	$\mathcal{M}$ at W_e
544579	32	16659486	27	0.2	1.000	27	617018	16659486
				0.25	1.033	27	617018	16659486
				0.3	1.032	32	544579	17426528
p93791								
$\mathcal{T}_{min}$	${}^1W_1$	$\mathcal{M}_{min}$	${}^2W_2$	α	$\mathcal{C}_{min}$	${}^3W_e$	$\mathcal{T}$ at W_e	$\mathcal{M}$ at W_e
503661	62	29399656	22	0.5	1.012	22	1336348	29399656
				0.95	1.000	62	503661	31226982
				0.99	1.000	62	503661	31226982

${}^1W_1$: W at $\mathcal{T}_{min}$; ${}^2W_2$: W at $\mathcal{M}_{min}$; ${}^3W_e$: W at $\mathcal{C}_{min}$

7. CONCLUSION

We have presented new techniques based on rectangle packing for wrapper/TAM co-optimization and test scheduling. TAMs have been tailored to the test data needs of cores through the use of Pareto-optimal widths. We have also presented several heuristics that minimize the idle time on TAM wires, thereby leading to a fast and efficient algorithm for TAM width allocation and test scheduling. The new rectangle packing algorithm is scalable for large industrial SOCs. Finally, the proposed approach allows the system integrator to determine an SOC-level TAM width to trade off testing time with tester data volume.

8. REFERENCES

[1] J. Aerts and E.J. Marinissen. Scan chain design for test time reduction in core-based ICs. *Proc. Int. Test Conf.*, pp. 448-457, 1998.

[2] V.D. Agrawal, C.R. Kime and K.K. Saluja. A Tutorial on Built-In Self-Test, Part 1: Principles. *IEEE Design & Test of Computers*, pp. 73–82, March 1993.

[3] C. Barnhart et al. OPMISR: The foundation for compressed ATPG vectors. *Proc. Int. Test Conf.*, pp. 748–757, 2001.

[4] K. Chakrabarty. Optimal test access architectures for system-on-a-chip. *ACM Trans. Design Automation of Electronic Systems*, vol. 6, pp. 26–49, January 2001.

[5] K. Chakrabarty. Test scheduling for core-based systems using mixed-integer linear programming. *IEEE Trans. CAD*, vol. 19, pp. 1163-1174, Oct 2000.

[6] A. Chandra and K. Chakrabarty. Test resource partitioning for SOCs. *IEEE Design & Test of Computers*, vol. 18, pp. 80-91, Sept-Oct 2001.

[7] R.M. Chou, K.K. Saluja and V.D. Agrawal. Scheduling tests for VLSI systems under power constraints. *IEEE Trans. VLSI Systems*, vol. 5, no. 2, June 1997.

[8] E.G. Coffman, Jr., M.R. Garey, D.S. Johnson, and R.E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM J. Computing*, vol. 9, pp. 809–826, 1980.

[9] Y. Huang et al. Resource allocation and test scheduling for concurrent test of core-based SOC design. *Proc. Asian Test Symp.*, pp. 265-270, 2001.

[10] Y. Huang et al. On concurrent test of core-based SOC design. *J. Electronic Testing: Theory and Applications*, vol. 18, Aug. 2002, to appear.

[11] V. Iyengar and K. Chakrabarty. Precedence-based, preemptive and power-constrained test scheduling for system-on-a-chip. *Proc. VLSI Test Symp.*, pp. 368–374, 2001.

[12] V. Iyengar, K. Chakrabarty, and E.J. Marinissen. Test wrapper and test access mechanism co-optimization for system-on-chip. *J. Electronic Testing: Theory and Applications*, vol. 18, pp. 213–230, April 2002.

[13] V. Iyengar, K. Chakrabarty, and E.J. Marinissen. Efficient wrapper/TAM co-optimization for large SOCs. *Proc. Design Automation and Test in Europe (DATE) Conf.*, pp. 491–498, 2002.

[14] V. Iyengar, K. Chakrabarty, and E.J. Marinissen. On using rectangle packing for SOC wrapper/TAM co-optimization. *Proc. VLSI Test Symp.*, 2002, to appear.

[15] W. Jiang and B. Vinnakota. Defect-oriented test scheduling. *Proc. VTS*, pp. 433–438, 1999.

[16] E. Larsson and Z. Peng. Test scheduling and scan-chain division under power constraint. *Proc. Asian Test Symp.*, pp. 259–264, 2001.

[17] E.J. Marinissen, V. Iyengar and K. Chakrabarty. ITC 2002 SOC test benchmark initiative. *http://www.extra.research.philips.com/itc02socbenchm*

[18] E.J. Marinissen. The role of test protocols in automated test generation for embedded-core-based system ICs. *J. Electronic Testing: Theory and Applications*, vol. 18, Aug. 2002, to appear.

[19] M. Nourani and C. Papachristou. An ILP formulation to optimize test access mechanism in system-on-chip testing. *Proc. Int. Test Conf.*, pp. 902–910, 2000.