JOURNAL OF ELECTRONIC TESTING: Theory and Applications 18, 213–230, 2002
© 2002 Kluwer Academic Publishers. Manufactured in The Netherlands.

Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip*

VIKRAM IYENGAR AND KRISHNENDU CHAKRABARTY
Department of Electrical and Computer Engineering, Duke University, 130 Hudson Hall, Box 90291, Durham, NC 27708, USA
vik@ee.duke.edu
krish@ee.duke.edu

ERIK JAN MARINISSEN
Philips Research Laboratories, IC Design—Digital Design and Test, Prof. Holstlaan 4, M/S WAY-41, 5656 AA Eindhoven, The Netherlands
Erik.Jan.Marinissen@philips.com

Received June 29, 2001; Revised August 7, 2001

Editor: C. Landrault

Abstract. Test access mechanisms (TAMs) and test wrappers are integral parts of a system-on-chip (SOC) test architecture. Prior research has concentrated on only one aspect of the TAM/wrapper design problem at a time, i.e., either optimizing the TAMs for a set of pre-designed wrappers, or optimizing the wrapper for a given TAM width. In this paper, we address a more general problem, that of carrying out TAM design and wrapper optimization in conjunction. We present an efficient algorithm to construct wrappers that reduce the testing time for cores. Our wrapper design algorithm improves on earlier approaches by also reducing the TAM width required to achieve these lower testing times. We present new mathematical models for TAM optimization that use the core testing time values calculated by our wrapper design algorithm. We further present a new enumerative method for TAM optimization that reduces execution time significantly when the number of TAMs being designed is small. Experimental results are presented for an academic SOC as well as an industrial SOC.

Keywords: Embedded core testing, test access mechanism (TAM), test wrapper, testing time, integer linear programming

1. Introduction

System-on-chip (SOC) integrated circuits composed of processors, memories, and peripheral interface devices in the form of embedded cores, are now commonplace. Nevertheless, there remain several roadblocks to rapid and efficient system integration. Test development is now seen as a major bottleneck in SOC design, and test challenges are a major contributor to the widening gap between design and manufacturing capability [22].

The 1999 International Technology Roadmap for Semiconductors [18] clearly identifies test access for SOC cores as one of the key challenges for the near future. Test access mechanisms (TAMs) and test wrappers have been proposed as important components of an SOC test access architecture [22]. TAMs deliver

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

pre-computed test sequences to cores on the SOC, while test wrappers translate these test sequences into patterns that can be applied directly to the cores.

Test wrapper and TAM design is of critical importance in SOC system integration since it directly impacts the vector memory depth required on the ATE, as well as testing time, and thereby affects test cost. A TAM and wrapper design that minimizes the idle time spent by TAMs and wrappers during test directly reduces the number of don't-care bits in vectors stored on the tester, thereby reducing vector memory depth. The design of efficient test access architectures has become an important focus of research in core test integration [1, 3–6, 12, 15, 17]. This is especially timely and relevant, since the proposed IEEE P1500 standard provides a lot of freedom in optimizing its standardized, but scalable wrapper, and leaves TAM optimization entirely to the system integrator [10, 16].

The general problem of SOC test integration includes the design of TAM architectures, optimization of core wrappers, and test scheduling. The goal is to minimize the testing time, area costs, and power consumption during testing. The wrapper/TAM co-optimization problem that we address in this paper is as follows. Given the test set parameters for the cores on the SOC, as well as the total TAM width, determine an optimal number of TAMs for the SOC, an optimal partition of the total TAM width among the TAMs, an optimal assignment of cores to each TAM, and an optimal wrapper design for each core, such that the overall system testing time is minimized. In order to solve this problem, we examine a progression of three incremental problems structured in order of increasing complexity, such that they serve as stepping-stones to the more general problem of wrapper/TAM design. The first problem $\mathcal{P}_{\mathbf{W}}$ is related to test wrapper design. The next two problems $\mathcal{P}_{\mathbf{AW}}$ and $\mathcal{P}_{\mathbf{PAW}}$ are related to wrapper/TAM co-optimization.

1. $\mathcal{P}_{\mathbf{W}}$: Design a *wrapper* for a given core, such that (a) the core testing time is minimized, and (b) the TAM width required for the core is minimized.
2. $\mathcal{P}_{\mathbf{AW}}$: Determine (i) an *assignment* of cores to TAMs of given widths and (ii) a *wrapper* design for each core, such that SOC testing time is minimized. (Item (ii) equals $\mathcal{P}_{\mathbf{W}}$.)
3. $\mathcal{P}_{\mathbf{PAW}}$: Determine (i) a *partition* of the total TAM width among the given number of TAMs, (ii) an *assignment* of cores to the TAMs, and (iii) a *wrapper* design for each core, such that SOC testing time is minimized. (Items (ii) and (iii) together equal $\mathcal{P}_{\mathbf{AW}}$.)

These three problems lead up to $\mathcal{P}_{\mathbf{NPAW}}$, the more general problem of wrapper/TAM co-optimization described as follows.

$\mathcal{P}_{\mathbf{NPAW}}$: Determine (i) the *number* of TAMs for the SOC, (ii) a *partition* of the total TAM width among the TAMs, (iii) an *assignment* of cores to TAMs, and (iv) a *wrapper* design for each core, such that SOC testing time is minimized. (Items (ii), (iii) and (iv) together equal $\mathcal{P}_{\mathbf{PAW}}$.)

In the remainder of this paper, we formally define and analyze these problems, and propose solutions for each.

In this paper, we assume the "test bus" model for TAMs. We assume that the TAMs on the SOC operate independently of each other; however, the cores on a single TAM are tested sequentially. This can be implemented either by (i) multiplexing all the cores assigned to a TAM, or (ii) by testing one of the cores on the TAM, while the other cores on the TAM are bypassed. Furthermore, in this paper, we are addressing the problem of core test only; hence, we do not discuss issues related to test wrapper bypass and interconnect test.

The organization of this paper is as follows. In Section 2, we discuss prior work in the area of TAM and test wrapper design. In Section 3, we present two SOCs that we use as running examples throughout the paper. In Section 4, we address Problem $\mathcal{P}_{\mathbf{W}}$. In Section 5, we develop improved integer linear programming (ILP) models for optimizing core assignment to TAMs (Problem $\mathcal{P}_{\mathbf{AW}}$). In Section 6, we present new ILP models for TAM width partitioning (Problem $\mathcal{P}_{\mathbf{PAW}}$). In Section 7, we present an enumerative method that can often reduce the execution time required to solve $\mathcal{P}_{\mathbf{PAW}}$ when the number of TAMs is small. Finally, in Section 8, we examine $\mathcal{P}_{\mathbf{NPAW}}$, the general problem of wrapper/TAM co-optimization. Section 9 concludes the paper.

2. Prior Work

Test wrappers provide a variety of operation modes, including normal operation, core test, interconnect test, and (optional) bypass [14]. In addition, test wrappers need to be able to perform test width adaptation if the width of the TAM is not equal to the number of core terminals. The IEEE P1500 standard addresses the design of a flexible, scalable wrapper to allow modular testing [10, 16]. This wrapper is flexible and can be optimized to suit the type of TAM and test requirements for the core.

A "test collar" was proposed in [20] to be used as a test wrapper for cores. However, test width adaptation and interconnect test were not addressed. The issue of efficient de-serialization of test data by the use of balanced wrapper scan chains was discussed in [6]. Balanced wrapper scan chains, consisting of chains of core I/Os and internal scan chains, are desirable because they minimize the time required to scan in test patterns from the TAM. However, no mention was made of the method to be used to arrive at a balanced assignment of core I/Os and internal scan chains to TAM lines. The TESTSHELL proposed in [14] has provisions for the IEEE P1500 required modes of operation. Furthermore, heuristics for designing balanced wrapper scan chains, based on approximation algorithms for the well-known Bin Design problem [7], were presented in [15]. However the issue of reducing the TAM width required for a wrapper was not addressed.

A number of TAM designs have been proposed in the literature. These include multiplexed access [11], partial isolation rings [19], core transparency [8], dedicated test bus [20], reuse of the existing system bus [9], and a scalable bus-based architecture called TEST-RAIL [14]. Bus-based TAMs, being flexible and scalable, appear to be the most promising. However, their design has largely been ad hoc and previous methods have seldom addressed the problem of minimizing testing time under TAM width constraints. While [1] presents several novel TAM architectures (i.e., multiplexing, daisy chaining and distribution), it does not directly address the problem of optimal sizing of TAMs in the SOC. In particular, only internal scan chains are considered in [1], while wrappers and functional I/Os are ignored. Moreover, the lengths of the internal scan chains are not considered fixed, and therefore [1] does not directly address the problem of designing test architectures for hard cores.

More recently, integrated TAM design and test scheduling has been attempted in [13, 18]. However, in [13, 17], the problem of optimizing test bus widths and arbitrating contention among cores for test width was not addressed. In [17], the cost estimation for TAMs was based on the number of bridges and multiplexers used; the number of TAM wires was not taken into consideration. Furthermore, in [12] the impact of TAM widths on testing time was not included in the cost function. The relationship between testing time and TAM widths using ILP was examined in [3, 5], and TAM width optimization under power and routing constraints was studied in [4]. However, the problem of effective test width adaptation in the wrapper was not addressed. This led to an overestimation of testing time and TAM width. Improved wrapper designs and new ILP models for TAM design are therefore necessary.

In this paper, we present a new wrapper/TAM co-optimization methodology that overcomes the limitations of previous TAM design approaches that have addressed TAM optimization and wrapper design as independent problems. The new wrapper design algorithm that we present improves upon previous approaches by minimizing the core testing time, as well as reducing the TAM width required for the core. We propose an approach based on ILP to solve the problems of determining an optimal partition of the total TAM width and determining an optimal core assignment to the TAMs. We also address a new problem, that of determining the optimal number of TAMs for an SOC. This problem gains importance with increasing SOC size. This paper, to the best of our knowledge, is the first in which a wrapper/TAM co-optimization methodology has been applied to an industrial SOC.

3. Example SOCs

In order to illustrate the proposed wrapper/TAM co-optimization methods presented in this paper, and to demonstrate their effectiveness, we use two representative SOCs as running examples throughout this paper. The first one is an academic SOC named d695 (described as system $\mathcal{S}_2$ in [5]), and the second one is an industrial SOC from Philips, named p93791. The number (e.g., 93791) in each SOC name is a measure of its test complexity. This number is calculated by considering the numbers of functional inputs n_i, functional outputs m_i, bidirectional I/Os b_i, scan chains sc_i, internal scan chain lengths $l_{i,1}, l_{i,2}, \dots, l_{i,sc_i}$, and test patterns p_i for each Core i, as well as the total number of cores N in the SOC, all of which contribute to the complexity of the wrapper/TAM co-optimization problem. We calculate the SOC test complexity number using the formula $N \cdot \sum_{i=1}^{N} p_i \cdot (n_i + m_i + b_i + \sum_{r=1}^{sc_i} l_{i,r}) \cdot \frac{1}{10000}$. The letters "d" and "p" in d695 and p93791 refer to Duke University and Philips, respectively.

SOC d695 consists of two combinational ISCAS'85 and eight sequential ISCAS'89 benchmark circuits. Table 1 presents the test data for each core in d695. We assume that the ISCAS'89 circuits contain well-balanced internal scan chains. The proposed wrapper/TAM co-optimization methodology is also applicable

Table 1. Test data for the cores in d695 [5].

Circuit (core)	No. of Test patterns	No. of Primary I/Os	No. of Scan chains	Scan chain lengths	
				Min	Max
c6288	12	64	–	–	–
c7552	73	315	–	–	–
s838	75	35	1	32	32
s9234	105	75	4	52	54
s38584	110	342	32	44	45
s13207	234	214	16	39	41
s15850	95	227	16	33	34
s5378	97	84	4	44	46
s35932	12	355	32	54	54
s38417	68	134	32	51	55

to SOCs containing non-scanned sequential cores, since these cores can be treated as combinational (having zero-length internal scan chains) for the purpose of testing time calculation. SOC p93791 contains 32 cores. Of these, 18 are memory cores embedded within hierarchical logic cores. Table 2 presents the data for the 14 logic cores and 18 embedded memories in SOC p93791.

The experimental results presented in this paper were obtained using a Sun Ultra 80 with a 450 MHz processor and 4096 MB memory.

4. Test Wrapper Design

A standardized, but scalable test wrapper is an integral part of the IEEE P1500 working group proposal [10]. A test wrapper is a layer of DfT logic that connects a TAM to a core for the purpose of test [22]. Test wrappers have four main modes of operation. These are (i) *Normal* operation, (ii) *Intest*: core-internal testing, (iii) *Extest*: core-external testing, i.e., interconnect test, and (iv) *Bypass* mode. Wrappers may need to perform test width adaptation when the TAM width is not equal to the number of core terminals. This will often be required in practice, since large cores typically have hundreds of core terminals, while the total TAM width is limited by the number of SOC pins. In this paper, we address the problem of TAM design for *Intest*, and therefore we do not discuss issues related to *Bypass* and *Extest*.

A core usually contains several core I/Os as well as several internal scan chains consisting of flip-flops connected in serial within the core for the purpose of scanning test data in and out of the core. To perform test width adaptation, wrapper scan chains are constructed by connecting core I/Os and internal scan chains in serial. The number of wrapper scan chains constructed is equal to the TAM width provided to the core; hence each wrapper scan chain is assigned to a single unique TAM line. Thus the test data width (number of core terminals) of the core is adapted to the TAM width.

Table 2. Test data for the cores in p93791.

Core	No. of Test patterns	No. of Primary I/Os	No. of Scan chains	Scan chain lengths	
				Min	Max
Core 1	409	213	46	1	168
Core 2	192	74	–	–	–
Core 3	648	69	–	–	–
Core 4	11	117	23	4	5
Core 5	6127	248	–	–	–
Core 6	218	813	46	500	521
Core 7	177	41	–	–	–
Core 8	177	41	–	–	–
Core 9	192	77	–	–	–
Core 10	1164	395	–	–	–
Core 11	187	286	11	17	82
Core 12	391	369	46	92	93
Core 13	194	214	46	173	219
Core 14	194	214	46	173	219
Core 15	288	78	–	–	–
Core 16	396	201	–	–	–
Core 17	216	283	43	145	150
Core 18	42	113	–	–	–
Core 19	210	903	44	97	100
Core 20	416	220	44	132	181
Core 21	42	113	–	–	–
Core 22	42	76	–	–	–
Core 23	234	205	46	1	175
Core 24	3072	21	–	–	–
Core 25	2688	45	–	–	–
Core 26	96	76	–	–	–
Core 27	916	109	46	50	68
Core 28	396	159	–	–	–
Core 29	172	231	35	185	189
Core 30	192	77	–	–	–
Core 31	204	218	–	–	–
Core 32	3084	396	–	–	–

The problem of designing an effective width adaptation mechanism for *Intest* can be broken down into three problems [15]: (i) partitioning the set of wrapper scan chain elements (internal scan chains and wrapper cells) into several wrapper scan chains, which are equal in number to the number of TAM lines, (ii) ordering the scan elements on each wrapper chain, and (iii) providing optional bypass paths across the core. The problems of ordering scan elements on wrapper scan chains and providing bypass paths were shown to be simple in [15], while that of partitioning wrapper scan chain elements was shown to be $\mathcal{NP}$-hard. Therefore, in this section, we address only the problem of effectively partitioning wrapper scan chain elements into wrapper scan chains.

Recent research on wrapper design has stressed the need for balanced wrapper scan chains [6, 15]. *Balanced* wrapper scan chains are those that are as equal in length to each other as possible. Balanced wrapper scan chains are important because the number of clock cycles to scan in (out) a test pattern to (from) a core is a function of the length of the longest wrapper scan-in (scan-out) chain. Let s_i (s_o) be the length of the longest wrapper scan-in (scan-out) chain for a core. The time required to apply the entire test set to the core is then given by $T = (1 + \max\{s_i, s_o\}) \cdot p + \min\{s_i, s_o\}$, where p is the number of test patterns. This time T decreases as both s_i and s_o are reduced, i.e., as the wrapper scan-in (and scan-out) chains become more equal in length.

Fig. 1 illustrates the difference between balanced and unbalanced wrapper scan chains; *Bypass* and *Extest* mechanisms are not shown. In Fig. 1(a), wrapper scan chain 1 consists of two input cells and two output cells, while wrapper scan chain 2 consists of three internal scan chains that contain 14 flip-flops in total. This results in unbalanced wrapper scan-in/out chains and a scan-in and scan-out time per test pattern of 14 clock cycles each. On the other hand, with the same elements and TAM width, the wrapper scan chains in Fig. 1(b) are balanced. The scan-in and scan-out time per test pattern is now 8 clock cycles.

The problem of partitioning wrapper scan chain elements into balanced wrapper scan chains was shown to be equivalent to the well-known Multiprocessor Scheduling and Bin Design problems in [15]. In this paper, the authors presented two heuristic algorithms for the Bin Design problem to solve the wrapper scan chain element partitioning problem. Given k TAM lines and sc internal scan chains, the authors assigned the scan elements to m wrapper scan chains, such that $\max\{s_i, s_o\}$ was minimized. This approach is effective if the goal is to minimize only $\max\{s_i, s_o\}$. However, we are addressing the wrapper design problem as part of the more general problem of wrapper/TAM co-optimization, and therefore we would also like to minimize the number of wrapper chains created. This can be explained as follows. Consider a core that has four internal scan chains of lengths 32, 8, 8, and 8, respectively, 4 functional inputs, and 2 functional outputs. Let the number of TAM lines provided be 4. The algorithms in [15] will partition the scan elements among four wrapper scan chains as shown in Fig. 2(a), giving $\max\{s_i, s_o\} = 32$. However, the scan elements may also be assigned to only 2 wrapper scan chains as shown in Fig. 2(b), which also gives $\max\{s_i, s_o\} = 32$. The second assignment, however, is clearly more efficient in terms of TAM width

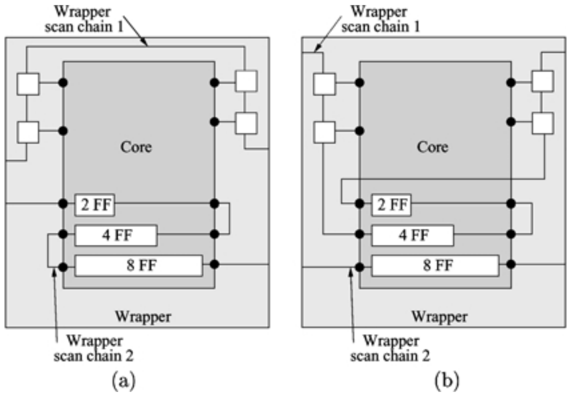

Fig. 1. Wrapper chains: (a) unbalanced, (b) balanced.

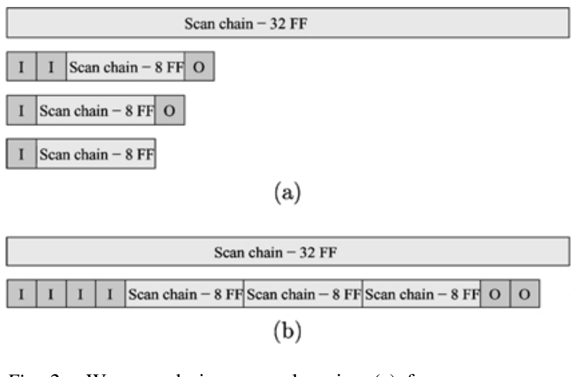

Fig. 2. Wrapper design example using (a) four wrapper scan chains, and (b) two wrapper scan chains.

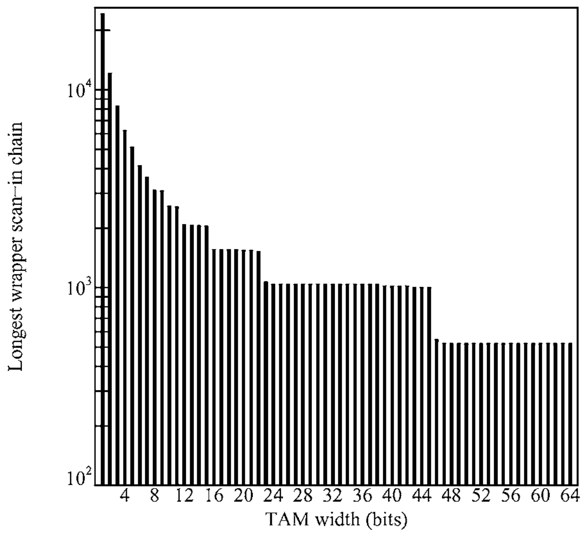

Fig. 3. Decrease in s_i with increasing k for Core 6 of p93791.

utilization, and therefore would be more useful for a wrapper/TAM co-optimization strategy.

Consider Core 6, the largest logic core from p93791. Core 6 has 417 functional inputs, 324 functional outputs, 72 bidirectional I/Os, and 46 internal scan chains of lengths: 7 scan chains $\times$ 500 bits, 30 scan chains $\times$ 520 bits, and 9 scan chains $\times$ 521 bits, respectively. The Combine algorithm [15] was used to create wrapper configurations for Core 6, for values of k between 1 and 64 bits, where k is the number of TAM lines provided to the core. Since the functional inputs in Core 6 outnumber the functional outputs, $\max\{s_i, s_o\} = s_i$. The value of s_i obtained for each value of k is illustrated in Fig. 3. From the graph, we observe that as k increases, s_i decreases in a series of distinct steps. This is because as k increases, the core internal scan chains are redistributed among a larger number of wrapper scan-in chains; thus s_i decreases only when the increase in k is sufficient to remove an internal scan chain from the longest wrapper scan-in chain. For example, when the internal scan chains in Core 6 are distributed among 24 wrapper scan-in chains, $s_i = 1040$ bits long. The value of s_i remains at 1040 until k reaches 39, when s_i drops to 1020. Hence, for $24 \le k \le 38$, only 24 wrapper scan-in chains need be designed. Our wrapper design strategy is therefore to (i) minimize testing time by minimizing $\max\{s_i, s_o\}$, and (ii) identify the maximum number k' of wrapper scan chains that actually need to be created to minimize testing time, when k TAM lines are provided to the wrapper. The set of values of k' corresponding to the values of $1 \le k \le \infty$ is known as the set of pareto-optimal points for the graph.

$\mathcal{P}_\mathrm{W}$, the two-priority wrapper optimization problem that this section addresses can now be formally stated as follows.

- $\mathcal{P}_\mathrm{W}$: Given a core with n functional inputs, m functional outputs, sc internal scan chains of lengths $l_1, l_2, \ldots, l_{sc}$, respectively, and TAM width k, assign the $n + m + sc$ wrapper scan chain elements to $k' \le k$ wrapper scan chains such that (i) $\max\{s_i, s_o\}$ is minimized, where s_i (s_o) is the length of the longest wrapper scan-in (scan-out) chain, and (ii) k' is minimum subject to priority (i).

Priority (ii) of $\mathcal{P}_\mathrm{W}$ is based on the earlier observation that $\max\{s_i, s_o\}$ can be minimized even when the number of wrapper scan chains designed is less than k. This reduces the width of the TAM required to connect to the wrapper. Problem $\mathcal{P}_\mathrm{W}$ is therefore analogous to the problem of Bin Design (minimizing the size of the bins), with the secondary priority of Bin Packing (minimizing the number of bins). If the value of k_{max} in Problem $\mathcal{P}_\mathrm{W}$ is always fixed at k, then Problem $\mathcal{P}_\mathrm{W}$ reduces to the Partitioning of Scan Chains (PSC) Problem [15], and is therefore clearly $\mathcal{NP}$-hard, since Problem PSC was shown to be $\mathcal{NP}$-hard in [15].

We have developed an approximation algorithm based on the Best Fit Decreasing (BFD) heuristic [7] to solve $\mathcal{P}_\mathrm{W}$ efficiently. The algorithm has three main parts, similar to [15]: (i) partition the internal scan chains among a minimal number of wrapper scan chains to minimize the longest wrapper scan chain length, (ii) assign the functional inputs to the wrapper scan chains created in part (i), and (iii) assign the functional outputs to the wrapper scan chains created in part (i). To solve part (i), the internal scan chains are sorted in descending order. Each internal scan chain is then successively assigned to the wrapper scan chain, whose length after this assignment is closest to, but not exceeding the length of the current longest wrapper scan chain. Intuitively, each internal scan chain is assigned to the wrapper scan chain in which it achieves the *best* fit. If there is no such wrapper scan chain available, then the internal scan chain is assigned to the current *shortest* wrapper scan chain. Next the process is repeated for part (ii) and part (iii), considering the functional inputs and outputs as internal scan chains of length 1. The pseudocode for our algorithm *Design_wrapper* is as follows.

Procedure *Design_wrapper*

```
Part (i)
1. Sort the internal scan chains in descending
   order of length
2. For each internal scan chain $l$
3.     Find wrapper scan chain $\mathcal{S}_{max}$ with current
       maximum length
4.     Find wrapper scan chain $\mathcal{S}_{min}$ with current
       minimum length
5.     Assign $l$ to wrapper scan chain $\mathcal{S}$, such that
       $\{\text{length}(\mathcal{S}_{max}) - (\text{length}(\mathcal{S}) + \text{length}(l))\}$
       is minimum
6.     If there is no such wrapper scan chain $\mathcal{S}$ then
7.         Assign $l$ to $\mathcal{S}_{min}$
Part (ii)
8. Repeat steps 1 through 7 to add the primary inputs
   to the wrapper chains created in part (i)
Part (iii)
9. Repeat steps 1 through 7 to add the primary inputs
   to the wrapper chains created in part (i)
```

We base our algorithm on the BFD heuristic mainly because BFD utilizes a more sophisticated partitioning rule than First Fit Decreasing (FFD), since each scan element is assigned to the wrapper scan chain in which it achieves the *best* fit [7]. FFD was used as a subroutine to the wrapper design algorithm in [15]. In our algorithm, a new wrapper scan chain is created only when it is not possible to fit an internal scan chain into one of the existing wrapper scan chains without exceeding the length of the current longest wrapper scan chain. Thus, while the algorithms presented in [15] always use k wrapper scan chains, *Design_wrapper* uses as few wrapper scan chains as possible, without compromising test application time. The worst-case complexity of the *Design_wrapper* algorithm is $O(sc \cdot \log sc + sc \cdot k)$, where sc is the number of internal scan chains and k is the limit on the number of wrapper scan chains.

We next present a wrapper design for the example Core A in [15]. Core A has 8 functional inputs `a[0:7]`, 11 functional outputs `z[0:10]`, 9 internal scan chains of lengths 12, 12, 8, 8, 8, 6, 6, 6, and 6 flip-flops, and a scan enable control `sc`. The test wrapper for A is to be connected to a 1-bit TAM `STP` (as mandated by the IEEE P1500 standard [10]) as well as to a 4-bit TAM `MTP[0:3]`. Furthermore, the test wrapper is to contain a bypass from TAM inputs to TAM outputs [16]. The *Design_wrapper* algorithm was used to design the wrapper for Core A. The scan elements in the core were partitioned among four wrapper scan chains using the *Design_wrapper* algorithm as shown in Table 3. This partition yields a longest scan-in chain of length 20, and a longest scan-out chain of length 21, both of which are optimal values for a 4-bit TAM. The wrapper designed for Core A is illustrated in Fig. 4.

From Fig. 3, we further observe that as k is increased beyond 47, there is no further decrease in testing time,

Table 3. Wrapper scan chains for Core A.

	Wrapper scan chains			
	1	2	3	4
Internal scan chains	$12+6$	$12+6$	$8+6+6$	$8+8$
Wrapper input cells	2	2	0	4
Wrapper output cells	3	3	0	5
Scan-in length	20	20	20	20
Scan-out length	21	21	20	21

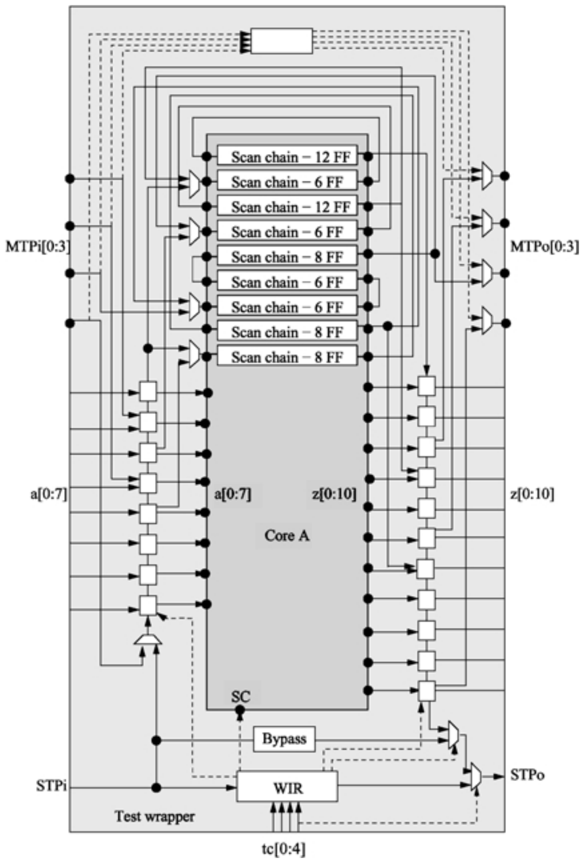

Fig. 4. Wrapper design for example Core A from [15].

since the longest internal scan chain of the core has been assigned to a dedicated wrapper scan chain, and there is no other wrapper scan chain longer than the longest internal scan chain. We next derive an expression for this maximum value of TAM width k_{max} required to minimize testing time for a core.

Theorem 1. *If a core has n functional inputs, m functional outputs, and sc internal scan chains of lengths* $l_1, l_2, \ldots, l_{sc}$, *respectively, an upper bound* k_{max} *on the TAM width required to minimize testing time is given by* $\lceil \frac{\max\{n,m\} + \sum_{i=1}^{sc} l_i}{\max_i\{l_i\}} \rceil$.

Proof: The test application time for a core is given by $T = (1 + \max\{s_i, s_o\}) \cdot p + \min\{s_i, s_o\}$, where s_i (s_o) is the length of the longest wrapper scan-in (scan-out) chain, and p is the number of test patterns in the test set. A lower bound on T, for any TAM width k, is therefore given by $(1 + \min_k\{\max\{s_i, s_o\}\}) \cdot p + \min_k\{\min\{s_i, s_o\}\}$. The lowest value that $\max\{s_i, s_o\}$ and $\min\{s_i, s_o\}$ can attain, is given by the length of the core's longest internal scan chain $\max_i\{l_i\}$. Therefore, $\min\{T\} = (1 + \max_i\{l_i\}) \cdot p + \max_i\{l_i\}$. Let the upper bound on k at which $\min\{T\}$ is reached be denoted as k_{max}. At this value of k, the number of flip-flops assigned to each wrapper scan chain (either scan-in or scan-out, whichever has more flip-flops) is at most $\max_i\{l_i\}$. Therefore k_{max} is the smallest integer, such that $k_{max} \cdot \max_i\{l_i\}$ is at least the sum of all the flip-flops on the wrapper scan chains, i.e., $k_{max} \cdot \max_i\{l_i\} \geq \max\{n, m\} + \sum_{i=1}^{sc} l_i$. Thus, k_{max} is the smallest integer, such that $k_{max} \geq \frac{\max\{n,m\} + \sum_{i=1}^{sc} l_i}{\max_i\{l_i\}}$. Therefore, $k_{max} = \lceil \frac{\max\{n,m\} + \sum_{i=1}^{sc} l_i}{\max_i\{l_i\}} \rceil$. ■

The value of k_{max} for each core can further be used to determine an upper bound on the TAM width for any TAM on the SOC. In Section 6, we show how k_{max} can be used to bound the TAM widths, when obtaining an optimal partition of total test width among TAMs on the SOC.

Table 4 presents results on the savings in TAM width obtained using *Design_wrapper* for Core 6. For larger values of k, the number of TAM lines actually used is far less than the number of available TAM lines; thus, with respect to TAM width utilization, *Design_wrapper* is considerably more efficient than the wrapper design algorithms proposed in [15].

In the next section, we address $\mathcal{P}_{\mathbf{AW}}$, the second problem of our TAM/wrapper co-optimization framework—determining an assignment of cores to TAMs of given widths and optimizing the wrapper design for each core.

Table 4. Savings in TAM width utilization for Core 6 of p93791.

Available TAM width	TAM width utilized	Length of longest wrapper scan chain
1	1	24278
2	2	12139
3	3	8263
4	4	6202
5	5	5142
6	6	4141
7	7	3621
8	8	3101
9	9	3081
10	10	2581
11	11	2561
12	12	2080
13	13	2061
14	14	2060
15	15	2041
16–19	16	1560
20–21	20	1540
22	22	1521
23	23	1056
24–38	24	1040
39–42	39	1020
43–45	43	1000
46	46	528
47–64	47	521

5. Optimal Core Assignment to TAMs

In this paper, we assume the "test bus" model for TAM design. We assume that each of the B TAMs on the SOC are independent; however, the cores on each TAM are tested in sequential order. This can be implemented either by (i) multiplexing all the cores assigned to a TAM as in Fig. 5(a), or (ii) by testing one of the cores on the TAM, while the other cores on the TAM are in *Bypass* mode as in Fig. 5(b). Furthermore, the core bypass may either be an internal bypass within the wrapper or an external bypass. This paper does not directly address the design of hierarchical TAMs. The SOC hierarchy is flattened for the purpose of TAM design and hierarchical cores are considered as being at the same level in test mode.

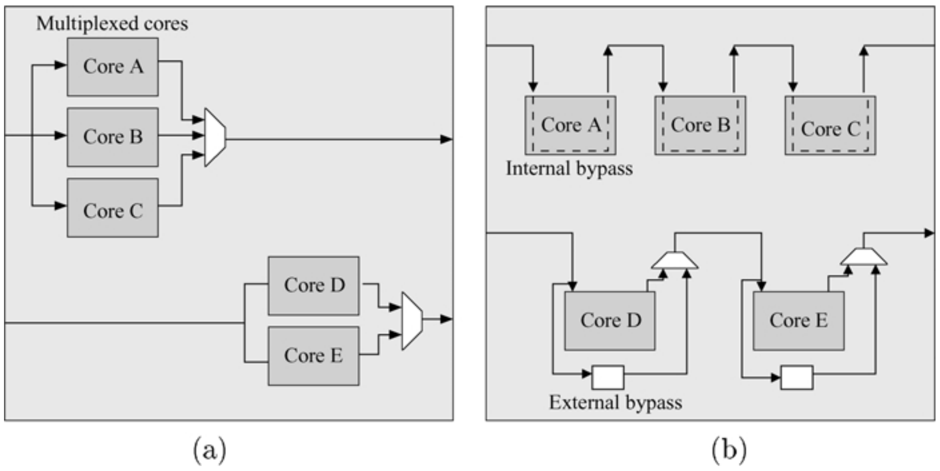

Fig. 5. Test bus model of TAM design: (a) multiplexed cores, (b) cores with bypass on a test bus.

The problem that we examine in this section, that of minimizing the system testing time by assigning cores to TAMs when TAM widths are known, can be stated as follows.

- $\mathcal{P}_{\mathbf{AW}}$: Given N cores and B TAMs of test widths $w_1, w_2, \ldots, w_B$, determine an assignment of cores to the TAMs and a wrapper design for each core, such that the testing time is minimized.

This problem can be shown to be $\mathcal{NP}$-hard using the techniques presented in [5]. However, for realistic SOCs the sizes of the problem instances were found to be small and could be solved exactly using an ILP formulation in execution times less than a second.

To model this problem, consider an SOC consisting of N cores and B TAMs of widths $w_1, w_2, \ldots, w_B$. If Core i is assigned to TAM j, let the time taken to test Core i be given by $T_i(w_j)$ clock cycles. The testing time $T_i(w_j)$ is calculated as $T_i(w_j) = (1 + \max\{s_i, s_o\}) \cdot p_i + \min\{s_i, s_o\}$, where p_i is the number of test patterns for Core i and s_i (s_o) is the length of the longest wrapper scan-in (scan-out) chain obtained from *Design_wrapper*. We introduce binary variables x_{ij} (where $1 \leq i \leq N$ and $1 \leq j \leq B$), which are used to determine the assignment of cores to TAMs in the SOC. Let x_{ij} be a 0-1 variable defined as follows:

$$x_{ij} = \begin{cases} 1, & \text{if Core } i \text{ is assigned to TAM } j \\ 0, & \text{otherwise} \end{cases}$$

The time needed to test all cores on TAM j is given by $\sum_{i=1}^{N} T_i(w_j) \cdot x_{ij}$. Since all the TAMs can be used simultaneously for testing, the system testing time equals $\max_{1 \leq j \leq B} \sum_{i=1}^{N} T_i(w_j) \cdot x_{ij}$.

A mathematical programming model for this problem can be formulated as follows.

Objective: Minimize $\mathcal{T} = \max_{1 \leq j \leq B} \sum_{i=1}^{N} T_i(w_j) \cdot x_{ij}$, subject to $\sum_{j=1}^{B} x_{ij} = 1$, $1 \leq i \leq N$, i.e., every core is connected to exactly one TAM.

Before we describe how a solution to $\mathcal{P}_{\mathbf{AW}}$ can be obtained, we briefly describe ILP, and then present the ILP formulation based on the above mathematical programming model to solve $\mathcal{P}_{\mathbf{AW}}$. The goal of ILP is to minimize a linear objective function on a set of integer variables, while satisfying a set of linear constraints [21]. A typical ILP model can described as follows:

$$\begin{aligned} &\text{minimize:} && \mathbf{Ax} \\ &\text{subject to:} && \mathbf{Bx} \leq \mathbf{C}, \quad \text{such that } \mathbf{x} \geq 0, \end{aligned}$$

where **A** is an objective vector, **B** is a constraint matrix, **C** is a column vector of constants, and **x** is a vector of integer variables. Efficient ILP solvers are now readily available, both commercially and in the public domain [2].

The minmax objective function of the mathematical programming model for $\mathcal{P}_{\mathbf{AW}}$ can be easily linearized to obtain the following ILP model.

Objective: Minimize $\mathcal{T}$, subject to

1. $\mathcal{T} \geq \sum_{i=1}^{N} T_i(w_j) \cdot x_{ij}$, $1 \leq j \leq B$, i.e., $\mathcal{T}$ is the maximum testing time on any TAM
2. $\sum_{j=1}^{B} x_{ij} = 1$, $1 \leq i \leq N$, i.e., every core is assigned to exactly one TAM

Table 5. Core assignment to TAMs of given widths for SOC d695.

TAM widths	TAM assignment	Testing time (clock cycles)
$8+16$	(2,1,1,1,2,2,2,1,2,1)	30086
$8+16+32$	(2,3,1,1,3,2,3,1,2,3)	14016
$8+12+16+20$	(2,3,1,1,3,4,4,1,2,2)	13323
$3+5+17+19+23$	(4,5,1,2,5,4,3,1,2,3)	11228
$4+7+16+17+28+31$	(4,4,4,2,6,5,4,1,4,3)	9878
$5+7+11+24+27+34+37$	(7,6,3,7,6,5,6,4,1,7)	9869

We solved this simple ILP model to determine optimal assignments of cores to TAMs for the SOCs introduced in Section 3. The number of variables and constraints for this model (a measure of the complexity of the problem) is given by NB, and $N + B = \mathcal{O}(N)$, respectively. The user time was less than a second in all cases. The optimal assignments of cores to TAMs of given widths for SOC d695 are shown in Table 5. Note that the testing times shown are optimal only for the *given* TAM widths; lower testing times can be achieved if an *optimal* TAM width partition is chosen. For example, Table 6 in Section 6 shows that a testing time lower than 29451 cycles can be achieved using two TAMs, if an optimal TAM width partition is chosen. In Sections 6 and 7, we will address the problem of determining optimal width partitions.

Lower Bounds on System Testing Time. For an SOC with N cores and B TAMs of widths $w_1, w_2, \ldots, w_B$, respectively, a lower bound on the total testing time $\mathcal{T}$ is given by $\max_i\{T_i(k)\}$, where $k = \max_j\{w_j\}$. The testing time for Core i depends on the width of the test bus to which it is assigned. Clearly, the testing time for Core i is at least $\min_j\{T_i(w_j)\}$. Since the overall system testing time is constrained by the core that has the largest test time, therefore $\mathcal{T} \geq \max_i\{T_i(k)\}$, where $k = \max_j\{w_j\}$. Intuitively, this value is the time needed to test the core that has the largest testing time when assigned to the widest TAM. For SOC d695 with two TAMs of 32 bits and 16 bits, respectively, the lower bound on the testing time is 6215 cycles. This corresponds to the testing time needed for Core 5 if it is assigned to TAM 1.

A lower bound on system testing time that does not depend on the given TAM widths can further be determined. This bound is related to the length of the longest internal scan chain of each core. The lower bound becomes tighter as we increase the number of TAMs. From Theorem 1, we know that for a Core i, where Core i has sc_i internal scan chains of lengths $l_{i,1}, l_{i,2}, \ldots, l_{i,sc_i}$, respectively, and the test set for Core i has p_i test patterns, a lower bound on the testing time is given by $(1 + \max_r\{l_{i,r}\}) \cdot p_i + \max_r\{l_{i,r}\}$. Therefore, for an SOC with N cores, a lower bound on the system testing time is given by $\max_i\{(1 + \max_r\{l_{i,r}\}) \cdot p + \max_r\{l_{i,r}\}\}$. Intuitively, this means that the system testing time is lower bounded by the time required to test the core with the largest testing time.

Table 6. Testing time for d695 obtained for $B = 2$.

	Results in [5]		Current wrapper/TAM co-optimization			
Total TAM width	Partition $w_1 + w_2$	Testing time	Partition $w_1 + w_2$	Testing time	Core assignment	Execution time (min)
16	$15+1$	2423712	$6+10$	45055	(1,2,1,1,2,1,2,1,2,2)	0.7
20	$19+1$	2363126	$4+16$	34444	(1,2,1,1,2,2,1,1,2,2)	0.8
24	$19+5$	2278443	$6+18$	29501	(1,2,1,1,2,2,1,1,1,2)	2.1
28	–[a]	–[a]	$8+20$	26964	(1,1,1,1,2,2,1,1,1,2)	3.9
32	$3+29$	2202286	$11+21$	25442	(1,2,1,1,2,2,2,1,2,1)	5.23
36	$4+32$	2174501	$16+20$	23312	(1,1,1,1,2,2,2,1,1,1)	11.0
40	$9+31$	2149720	$8+32$	21359	(1,2,1,1,2,2,1,1,2,2)	12.5
44	$12+32$	2123437	$10+34$	20883	(1,2,1,1,2,2,1,1,2,2)	13.0
48	$32+16$	2099390	$16+32$	19938	(1,1,1,1,2,2,1,1,1,2)	32.1
52	$32+20$	2086542	$20+32$	19087	(2,1,1,1,2,1,2,2,2,2)	50.1
56	$25+31$	2069738	$19+37$	18434	(2,2,1,1,2,1,2,2,2,2)	52.8
60	$28+32$	2044346	$20+40$	18205	(2,2,1,1,2,1,2,2,2,2)	76.7
64	$32+32$	2029753	$20+44$	18205	(2,2,1,1,2,1,2,2,2,2)	158.7

[a]Not reported in [5].

6. Optimal Partitioning of TAM Width

In this section, we address $\mathcal{P}_{\mathbf{PAW}}$: the problem of determining (i) optimal widths of TAMs, and (ii) optimal assignments of cores to TAMs, in conjunction with wrapper design. This is a generalization of the core assignment problem $\mathcal{P}_{\mathbf{AW}}$. We describe how testing times computed using the *Design_wrapper* algorithm in Section 4 are used to design the TAM architecture.

We assume that the total system TAM width can be at most W. From Theorem 1 in Section 4, we know that the width of each TAM need not exceed the maximum value of upper bound k_{max} for any core on the SOC. We denote this upper bound on the width of an individual TAM as w_{max}. For TAMs wider than w_{max}, there is no further decrease in testing time. Problem $\mathcal{P}_{\mathbf{PAW}}$ of minimizing testing time by optimal allocation of width among the TAMs can now be stated as follows.

- $\mathcal{P}_{\mathbf{PAW}}$: Given an SOC having N cores and B TAMs of total width W, determine a partition of W among the B TAMs, an assignment of cores to TAMs, and a wrapper design for each core, such that the total testing time is minimized.

This problem can be shown to be $\mathcal{NP}$-hard using the techniques presented in [5]. However, for many realistic SOCs, including p93791, the problem instance size is reasonable and can be solved exactly using an ILP formulation. This is also because the complexity of our solutions is related to the number of cores on the SOC and the numbers of their I/O ports and scan chains, and not the number of transistors or nets on the chip. A mathematical programming model for $\mathcal{P}_{\mathbf{PAW}}$ is shown below.

Objective: Minimize $\mathcal{T} = \max_j\{\sum_{i=1}^{N} T_i(w_j) \cdot x_{ij}\}$, subject to

1. $\sum_{j=1}^{B} x_{ij} = 1,\ 1 \le i \le N$, i.e., every core is connected to exactly one TAM
2. $\sum_{j=1}^{B} w_j = W, 1 \le j \le B$, i.e., the sum of all TAM widths is W
3. $w_j \le w_{max},\ 1 \le j \le B$, i.e., each TAM is at most w_{max} bits wide

The objective function and constraints of this mathematical programming model must now be linearized in order to express them in the form of an ILP model that can be solved by an ILP solver. We first express $T_i(w_j)$ as a sum: $T_i(w_j) = \sum_{k=1}^{w_{max}} \delta_{jk} \cdot T_i(k)$, by adding new binary indicator variables δ_{jk} (where $1 \le j \le B$, $1 \le k \le w_{max}$) to the mathematical programming model, such that:

$$\delta_{jk} = \begin{cases} 1, & \text{if TAM } j \text{ is } k \text{ bits wide} \\ 0, & \text{otherwise} \end{cases}$$

In addition, the following constraints are included in the model:

1. $w_j = \sum_{k=1}^{w_{max}} k \cdot \delta_{jk}$, i.e., a TAM can have values of width between 1 and w_{max}
2. $\sum_{k=1}^{w_{max}} \delta_{jk} = 1$, i.e., a TAM can have only one width

Intuitively, for every TAM j there is exactly one value of k for which δ_{jk} equals 1; therefore, the new indicator variables determine the width w_j of each TAM. The objective function now becomes $\mathcal{T} = \max_j \{\sum_{i=1}^{N} \sum_{k=1}^{w_{max}} \delta_{jk} \cdot T_i(k) \cdot x_{ij}\}$. The testing time $T_i(k)$ for various values of TAM width k can be efficiently calculated using the *Design_Wrapper* algorithm as shown in Section 4, and stored in the form of a look-up table for reference by the ILP solver.

Finally, the non-linear term $\delta_{jk} \cdot x_{ij}$ in the objective function can be linearized by replacing it with the variable y_{ijk} and the following two constraints:

1. $x_{ij} + \delta_{jk} - y_{ijk} \le 1$
2. $x_{ij} + \delta_{jk} - 2 \cdot y_{ijk} \ge 0$.

This is explained as follows. Consider first the case when $x_{ij} = 0$. From Constraints 1 and 2, we have $y_{ijk} + 1 \ge \delta_{jk}$ and $2 \cdot y_{ijk} \le \delta_{jk}$; since $\delta_{jk} \le 1$, therefore, y_{ijk} must equal 0. When $x_{ij} = 1$, we have $y_{ijk} \ge \delta_{jk}$ and $\delta_{jk} + 1 \ge 2 \cdot y_{ijk}$; therefore, $y_{ijk} = \delta_{jk}$.

The new variables and constraints yield the following ILP formulation.

Objective: Minimize $\mathcal{T} = \max_j\{\sum_{i=1}^{N} \sum_{k=1}^{w_{max}} y_{ijk} \cdot T_i(k)\}$, subject to

1. $x_{ij} + \delta_{jk} - y_{ijk} \le 1$
2. $x_{ij} + \delta_{jk} - 2 \cdot y_{ijk} \ge 0$
3. $w_j = \sum_{k=1}^{w_{max}} k \cdot \delta_{jk}$
4. $\sum_{k=1}^{w_{max}} \delta_{jk} = 1$
5. $\sum_{j=1}^{B} x_{ij} = 1$
6. $\sum_{j=1}^{B} w_j \le W$

The number of variables and constraints for these ILP models is given by $B \cdot (3N + w_{max} + 2) = \mathcal{O}(NB)$ (since w_{max} is a constant), and $B \cdot (5N + 5) + N = \mathcal{O}(NB)$, respectively. We solved the ILP model for

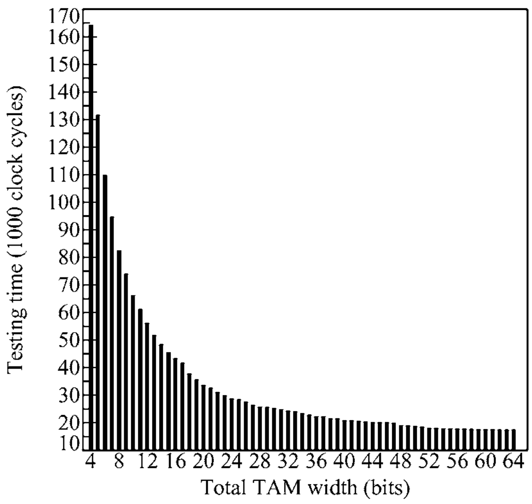

Fig. 6. Testing time for d695 for $B = 2$.

$\mathcal{P}_{\mathbf{PAW}}$ for several values of W and B. Table 6 and Fig. 6 present the values of testing time for SOC d695 obtained with two TAMs. The total TAM width partition among the two TAMs is shown and we also compare the testing times obtained with the testing times obtained in [5]. The testing time using the new wrapper design is at least an order of magnitude less than the time required in [5] for all cases. This was to be expected since an inefficient de-serialization model was used in [5]. The reductions in testing time diminish with increasing W. A pragmatic choice of W for the system might therefore be the point where the system testing time begins to level off. In Fig. 6, this occurs at $W = 24$.

Table 7 presents the values of testing time obtained with three TAMs. The testing times for $B = 3$ are lower than the values obtained for $B = 2$ in general. However, for $W \leq 14$, the testing time for $B = 3$ is more than that for $B = 2$ (not included in Table 7). This is because for small values of W, a larger number of TAMs makes the widths of individual TAMs very small. Once again, the testing time begins to level off, this time at $W = 32$; hence this is a good choice for trading off TAM width with testing time.

Table 8 presents the system testing times for SOC p93791 obtained using two TAMs. We halted the ILP solver after 1 hour for each value of W and tabulated the best results obtained. This was done to determine whether an efficient partition of TAM width and the corresponding testing time can be obtained using the ILP model within a reasonable execution time. In the next section, we present the optimal testing times obtained for p93791 using a new enumerative methodology, and show that the testing times obtained in Tables 7 and 8 using the ILP model for $\mathcal{P}_{\mathbf{PAW}}$ are indeed either optimal or close to optimal.

Table 7. Testing time for d695 obtained for $B = 3$.

Total TAM width	TAM width partition $w_1 + w_2 + w_3$	Core assignment	Testing time	Execution time (min)
16	$3+5+8$	(2,1,1,1,3,2,1,2,3,3)	42568	4.5
20	$2+2+16$	(2,3,2,2,3,3,1,2,3,3)	34444	16.0
24	$2+5+17$	(2,2,1,1,3,3,3,1,3,2)	28292	32.5
28	$4+8+16$	(2,3,1,1,2,3,3,1,1,3)	24812	52.3
32	$4+10+18$	(2,3,1,1,2,3,3,1,1,3)	21566	64.7
36	$4+16+16$	(2,2,1,1,2,3,2,1,2,3)	19564	85.0
40	$4+17+19$	(3,2,1,1,3,2,2,1,2,3)	17901	104.6
44	$4+18+22$	(3,2,1,1,2,3,2,1,2,3)	17051	180[a]
48	$4+18+26$	(2,2,1,1,3,2,2,1,2,3)	16984	180[a]
52	$16+32+4$	(3,2,1,3,2,1,2,3,2,2)	14852	180[a]
56	$16+34+6$	(3,2,3,3,2,1,2,3,1,2)	13637	180[a]
60	$6+20+34$	(2,2,1,1,3,2,3,1,2,3)	12987	180[a]
64	$6+20+38$	(2,2,1,1,3,2,3,1,2,3)	12941	180[a]

[a]*lpsolve* was halted after 180 minutes.

Table 8. Testing time for p93791 obtained with $B = 2$.

W	$w_1 + w_2$	Testing time
16	$9+7$	1811860
20	$10+10$	1526200
24	$1+23$	1239880
28	$9+19$	1119160
32	$9+23$	944881
36	$9+27$	924909
40	$9+31$	929848
44	$9+35$	873276
48	$9+39$	835526
52	$9+43$	807909
56	$9+47$	537891
60	$9+51$	545154
64	$9+55$	551111

Width Variant of Problem $\mathcal{P}_{\mathbf{PAW}}$. Closely related to $\mathcal{P}_{\mathbf{PAW}}$ is the problem of determining the minimum total TAM width W needed to satisfy a given testing time constraint. We call this problem the "width variant" of Problem $\mathcal{P}_{\mathbf{PAW}}$.

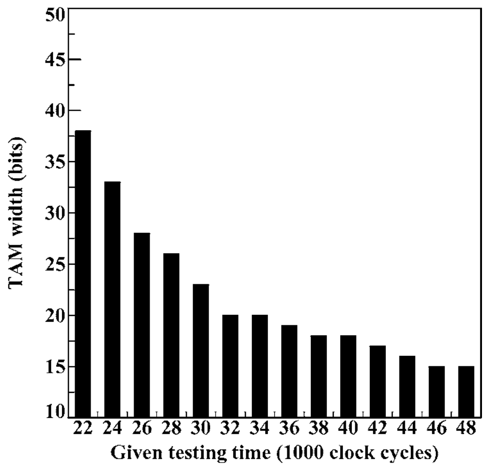

Fig. 7. The minimum TAM width needed to achieve a given testing time for d695 using two TAMs.

- $\mathcal{P}_{\mathbf{PAW}}$ width variant): Given N cores, B TAMs and a maximum testing time $\mathcal{T}$, determine the mimimum total TAM width W, an optimal partition of W among the B TAMs, an assignment of cores to TAMs, and test wrapper designs for cores such that the testing time is less than $\mathcal{T}$.

This problem has been shown to be $\mathcal{NP}$-hard as well [5]. However, an ILP formulation for the width variant of $\mathcal{P}_{\mathbf{PAW}}$ can be derived from the ILP model for $\mathcal{P}_{\mathbf{PAW}}$, simply by replacing the earlier objective function with:

Objective: Minimize $W = \sum_{j=1}^{B} w_j$ with the additional constraint:

$$\max_j \left\{ \sum_{i=1}^{N} \sum_{k=1}^{w_{max}} y_{ijk} \cdot T_i(k) \right\} \leq \mathcal{T}$$

Fig. 7 presents the minimum values of W needed to achieve a *given* testing time $\mathcal{T}$ for d695 using two TAMs. In our experiments, $\mathcal{T}$ was decreased from 48000 clock cycles to 22000 clock cycles, and the smallest value of W required to achieve a testing time lower than $\mathcal{T}$ was obtained.

7. Enumerative TAM Sizing

In Section 6, we showed that TAM optimization can be carried out using an ILP model for the $\mathcal{P}_{\mathbf{PAW}}$ problem. However, ILP is in itself an $\mathcal{NP}$-hard problem, and execution times can get high for large SOCs. A faster algorithm for TAM optimization that produces optimal results in short execution times is clearly needed. In Section 5, it was observed that the execution time of the model for Problem $\mathcal{P}_{\mathbf{AW}}$ was less than 1 second in all cases. We next demonstrate how the short execution time of this ILP model can be exploited to construct a series of $\mathcal{P}_{\mathbf{AW}}$ models that are solved to address the $\mathcal{P}_{\mathbf{PAW}}$ problem.

The pseudocode for an enumerative algorithm for $\mathcal{P}_{\mathbf{PAW}}$ that explicitly enumerates the unique partitions of W among the individual TAMs is as follows.

Procedure *P_{PAW}_enumerate*()

```
1. Let $W$ = total TAM width
2. Let $B$ = number of TAMs
3. While all unique partitions of $W$ have not been
   enumerated
4.   For TAM $j = 1$ to $B$
5.     For TAM width $w_j = 1$ to $\lfloor \frac{W - \sum_{k=0}^{j-1} w_k}{B-(j-1)} \rfloor$
6.         Create an ILP model for $\mathcal{P}_A$ for the TAM
           widths using Design_wrapper
7.         Determine core assignment and testing time
8.         Record the TAM design for the minimum
           testing time
```

The ILP models generated for each value of W in line 6 of *P_{PAW}_enumerate* are solved and the TAM width partition and core assignment delivering the best testing time are recorded. The solution obtained using *P_{PAW}_enumerate* is always optimal, because we generate all unique TAM width partitions, and then choose the solution with the lowest cost. Since lines 6 and 7 each take less than a second to execute, the execution time for *P_{PAW}_enumerate* is at most $2 \cdot p_B(W)$ seconds, where $p_B(W)$ is the number of partitions of W among B TAMs enumerated in line 3. The problem of determining the number of partitions $p_B(W)$ for a given choice of B TAMs can be addressed using partition theory in combinatorial mathematics [14]. In [14], $p_B(W)$ is shown to be approximately $\frac{W^{B-1}}{B!(B-1)!}$ for $W \mapsto \infty$. For $B = 2$, $p_2(W) = \lfloor \frac{W}{2} \rfloor$, since there are only $\lfloor \frac{W}{2} \rfloor$ unique ways of dividing an integer W into two smaller integers w_1 and w_2. Thus *P_{PAW}_enumerate* obtains the optimal solution for the $\mathcal{P}_{\mathbf{PAW}}$ problem for $W = 64$ and $B = 2$ in less than 64 seconds. For $B = 3$, the number of partitions work out to $p_3(W) = \sum_{i=0}^{\lfloor \frac{W}{3} \rfloor} \lfloor \frac{W-(3i+1)}{2} \rfloor$. From the above formula, the value of C for $W = 64$ and $B = 3$ is found to be 341. Therefore, the execution time of *P_{PAW}_enumerate* for $W = 64$ is upper bounded by 682 seconds or 11.6 minutes. This execution time is clearly reasonable, even for large W. In our experiments, we used a Sun Ultra 80, which solved the $\mathcal{P}_A$ models in well under a second of execution time. The time taken for P_{PAW}*_enumerate* was therefore significantly lower than the upper bound of $2 \cdot p_B(W)$ seconds even for large values of W.

We used P_{PAW}*_enumerate* to obtain the optimal TAM width assignment and minimal testing time for $B = 2$ and $B = 3$, for SOCs d695 and p93791. Results for SOC d695 are presented in Table 9. While the testing time for $B = 3$ is always less than the testing time for $B = 2$ and $W \geq 16$, the difference between $B = 2$ and $B = 3$ widens for larger W. This can be explained as follows. For smaller values of W, each individual TAM for $B = 3$ is narrow; hence, the testing time on each individual TAM increases sharply, as was observed earlier in Fig. 3.

Table 10 presents optimal results for enumerative TAM optimization for the p93791 SOC. Comparing these results with those presented in Table 8, we note that the results in Table 8 are indeed close to optimal. For example, for $W = 32$, the testing time presented in Table 8 is only 5% higher than optimal. Note that for both SOCs, the execution time for $\mathcal{P}_{\mathbf{AW}}$ is under 1 second. Hence similar execution times for P_{PAW}*_enumerate* are obtained for SOCs d695 and p93791. These execution times are significantly lower than those in Tables 6 and 7.

Table 9. Optimum testing time obtained for SOC d695 obtained using P_{PAW}*_enumerate*.

	B = 2			B = 3		
W	$w_1 + w_2$	$\mathcal{T}$	Exec. time (sec)	$w_1 + w_2 + w_3$	$\mathcal{T}$	Exec. time (sec)
16	6 + 10	45055	1	3 + 5 + 8	42568	4
20	4 + 16	34444	1	2 + 2 + 16	34444	6
24	6 + 18	29501	1	2 + 5 + 17	28292	10
28	8 + 20	26964	1	4 + 8 + 16	24812	12
32	11 + 21	25442	1	4 + 10 + 18	21566	16
36	16 + 20	23312	1	4 + 16 + 16	19564	21
40	8 + 32	21359	2	4 + 17 + 19	17901	24
44	10 + 34	20883	2	4 + 19 + 21	16975	30
48	16 + 32	19938	2	4,19,25	16975	40
52	20 + 32	19087	2	4,16,32	14852	44
56	19 + 37	18434	2	5,18,33	13207	53
60	20 + 44	18205	3	5 + 19 + 36	12941	63
64	20 + 44	18205	3	5 + 17 + 42	12941	84

Table 10. Optimal testing time obtained for SOC p93791 using P_{PAW}*_enumerate*.

	B = 2			B = 3		
W	$w_1 + w_2$	$\mathcal{T}$	Exec. time (sec)	$w_1 + w_2 + w_3$	$\mathcal{T}$	Exec. time (sec)
16	4 + 12	1798740	1	5 + 3 + 8	1771720	5
20	4 + 16	1448010	1	5 + 7 + 8	1426580	8
24	1 + 23	1211740	1	7 + 8 + 9	1187990	10
28	5 + 23	1020620	1	2 + 3 + 23	1017120	13
32	9 + 23	894342	1	5 + 4 + 23	887751	17
36	13 + 23	813054	2	9 + 4 + 23	789167	23
40	17 + 23	747378	2	6 + 12 + 23	698583	26
44	21 + 23	706349	2	5 + 16 + 23	653788	33
48	2 + 46	622199	2	9 + 16 + 23	599373	42
52	6 + 46	565456	2	16 + 13 + 23	561875	46
56	10 + 46	524203	3	10 + 23 + 23	514688	54
60	14 + 46	499725	3	13 + 23 + 24	499468	65
64	18 + 46	467424	3	18 + 23 + 23	460328	88

We compared the optimal testing times presented in Tables 9 and 10 with the testing times obtained using an *equal* partition of W among the B TAMs. The testing time using an optimal partition of W was significantly lower than that obtained using an equal partition for all values of W. For example, for $W = 64$ and $B = 2$, a partition of $(w_1, w_2) = (32, 32)$ provided a testing time of 611821 clock cycles, which is an increase of 28.6% over the testing time of 475598 clock cycles obtained using an optimal partition of $(w_1, w_2) = (18, 46)$.

The execution time of P_{PAW}*_enumerate* is smaller than that of the ILP model in Section 6 because the number of enumerations for two and three TAMs is reasonable. However, when TAM optimization is carried out for a larger number of TAMs that have a larger number of partitions of W, the ILP model for $\mathcal{P}_{\mathbf{PAW}}$ is likely to be more efficient in terms of execution time. In addition, the ILP model presented in Section 6 is likely to be more efficient when constraints arising from place-and-route and power issues are included in TAM optimization [4].

8. General Problem of Wrapper/TAM Co-Optimization

In the previous sections, we presented a series of problems in test wrapper and TAM design, each of which was a generalized version of the problem preceding it. In this section, we present $\mathcal{P}_{\mathbf{NPAW}}$, the more general problem of wrapper/TAM optimization that the problems of the preceding sections lead up to. We also show how solutions to the previous problems can be used to formulate a solution for this general problem.

The general problem can be stated as follows.

- $\mathcal{P}_{\mathbf{NPAW}}$: Given an SOC having N cores and a total TAM width W, determine the number of TAMs, a partition of W among the TAMs, an assignment of cores to TAMs, and a wrapper design for each core, such that the total testing time is minimized.

We use the method of restriction to prove that $\mathcal{P}_{\mathbf{NPAW}}$ is $\mathcal{NP}$-hard. We first define a new Problem $\mathcal{P}_{\mathbf{NPAW_1}}$, which consists of only those instances of $\mathcal{P}_{\mathbf{NPAW}}$ for which (i) $W = 2$, and (ii) all cores on the SOC have a single internal scan chain and no functional terminals. Hence, each core will have the same testing time on a 1-bit TAM as on a 2-bit TAM. An optimal solution to $\mathcal{P}_{\mathbf{NPAW_1}}$ will therefore always result in two TAMs of width one bit each. Problem $\mathcal{P}_{\mathbf{NPAW_1}}$ reduces to that of partitioning the set C of cores on the SOC into two subsets C_1 and $C - C_1$, such that each subset is assigned to a separate 1-bit TAM, and the difference between the sum of the testing times of the cores (on the first 1-bit TAM) and the sum of the testing times of the cores (on the 2nd 1-bit TAM) is minimized. Formally, the optimization cost function for $\mathcal{P}_{\mathbf{NPAW_1}}$ can be written as:

Objective: Minimize $\sum_{c \in C_1} T_c - \sum_{c \in C - C_1} T_c$, where T_c is the testing time of core c on a 1-bit TAM.

Next, consider the Partition problem [7], whose optimization variant can be stated as follows.

- **Partition**: Given a finite set A and a size $s(a) \in Z^+$ for each element $a \in A$, determine a partition of A into two subsets A_1 and $A - A_1$, such that $\sum_{a \in A_1} s(a) - \sum_{a \in A - A_1} s(a)$ is minimized.

That Problem $\mathcal{P}_{\mathbf{NPAW_1}}$ is equivalent to Partition can be established by the following four mappings: (i) $C \equiv A$, (ii) $C_1 \equiv A_1$, (iii) $T_c \equiv s(a)$, and (iv) $\sum_{c \in C_1} T_c \equiv \sum_{a \in A_1} s(a)$. Since Partition is known to be $\mathcal{NP}$-hard [7], $\mathcal{P}_{\mathbf{NPAW_1}}$ and $\mathcal{P}_{\mathbf{NPAW}}$ must also be $\mathcal{NP}$-hard.

To solve $\mathcal{P}_{\mathbf{NPAW}}$, we enumerate solutions for $\mathcal{P}_{\mathbf{PAW}}$ over several values of B. Our method for enumeration is outlined below.

Procedure *P_{NPAW}_enumerate*()

```
1. Let $W$ = total TAM width
2. Let $B_{MAX}$ = maximum number of TAMs
3. For $B = 2$ to $B_{MAX}$
4.     Execute Procedure $P_{PAW}$_enumerate
5.     Record the TAM design for the minimum
       testing time
```

For each value of W, the optimal number of TAMs, TAM width partition, core assignment, and wrapper designs for the cores are obtained. The solutions to $\mathcal{P}_{\mathbf{PAW}}$ for d695 for values of B ranging from 2 to 8 are illustrated in Fig. 8(a) and (b) for W values of 12, and 16 bits, respectively. In each figure, we observe that as B is increased from 2, the testing time decreases until a minimum value is reached at a particular value of B, after which the testing time stops decreasing and starts increasing as B is increased further. This is because for larger B, the width per TAM is small and testing time on each TAM increases significantly.

We next present a conjecture that formalizes the observation made in Fig. 8(a)–(c) and (d).

Conjecture 1. Let $T(S, W, B)$ denote the optimal testing time for SOC S having B TAMs and a total TAM width of W. If $T(S, W, B) \leq T(S, W, B+1)$, then $\forall_{X>B} T(S, W, B) \leq T(S, W, X)$.

We conjecture that during the execution of *P_{NPAW}_enumerate*, if at a certain value of B, the testing time is greater than or equal to the testing time at the previous value of B for the same total TAM width W, then the enumeration procedure can be halted and the optimal value of B recorded. Therefore, Conjecture 1 can be used to prune the search space for the optimal wrapper/TAM design. Since the execution time of *P_{NPAW}_enumerate* is particularly high for large values of B, we can achieve significant speed-ups in TAM optimization by halting the enumeration as soon as the minimum value of T is reached.

Based on Conjecture 1, we executed *P_{NPAW}_enumerate* for several values of W. In Table 11, we present the best testing times obtained for d695 for the values of W. For each value of W, the number of TAMs, width partition, testing time, and core assignment providing the minimum testing time is shown.

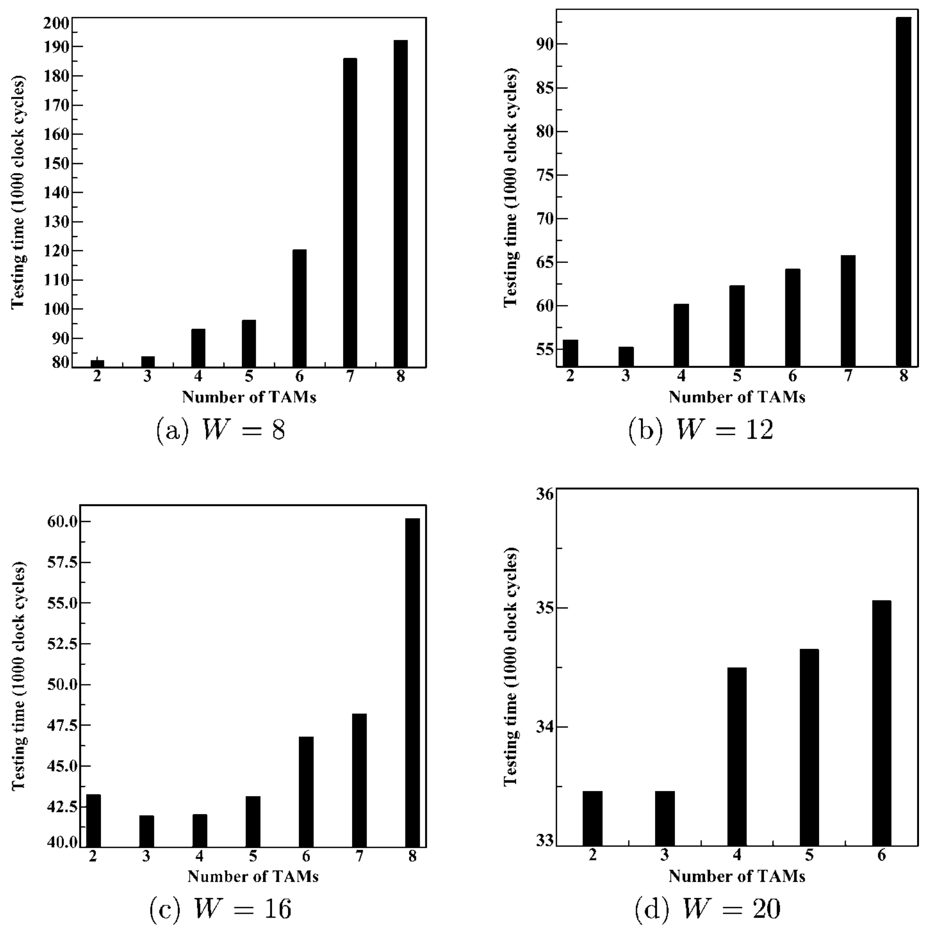

Fig. 8. Testing time for d695 obtained with increasing values of B. (a) $W = 8$; (b) $W = 12$; (c) $W = 16$; (d) $W = 20$.

9. Conclusion

We have investigated the problem of test wrapper/TAM co-optimization for SOCs, based on the test bus model of TAM design. In particular, we have formally defined the problem of determining the number of TAMs, a partition of the total TAM width among the TAMs, an assignment of cores to TAMs, and a wrapper design for each core, such that SOC testing time is minimized. To address this problem, we have formulated three incremental problems in test wrapper and TAM optimization that serve as stepping-stones to the more general problem stated above. We have proposed an efficient heuristic algorithm based on BFD for the wrapper design problem $\mathcal{P}_{\mathrm{W}}$ that minimizes testing time and TAM width. For $\mathcal{P}_{\mathrm{AW}}$, the problem of determining core assignments and wrapper designs, we have formulated an ILP model that results in optimal solutions in short execution times. We have formulated an ILP model to solve $\mathcal{P}_{\mathrm{PAW}}$, the problem of TAM width partioning that $\mathcal{P}_{\mathrm{AW}}$ leads up to. This ILP model was solved

Table 11. Best wrapper/TAM co-optimization results obtained for d695 for several values of W.

TAM width W	Optimum number of TAMs	Optimal width partition	Optimal core asignment	Optimum testing time
8	2	$4+4$	(1,1,1,1,1,2,1,1,2,1)	83580
12	3	$3+4+5$	(1,1,2,2,3,2,3,3,1,1)	56329
16	3	$3+5+8$	(2,1,1,1,3,2,1,2,3,3)	42568
20	3	$2+2+16$	(2,3,2,2,3,3,1,2,3,3)	34444
24	4	$1+2+9+12$	(2,4,2,1,4,3,3,2,2,4)	28289
28	4	$1+2+9+16$	(3,2,2,2,4,3,4,1,3,4)	23997

to obtain optimal TAM designs for reasonably-sized problem instances. We have also presented a new enumerative approach for $\mathcal{P}_{\mathbf{PAW}}$ that offers significant reductions in the execution time. Finally, we have defined a new wrapper/TAM design problem, $\mathcal{P}_{\mathbf{NPAW}}$, in which the number of TAMs to be designed must be determined. $\mathcal{P}_{\mathbf{NPAW}}$ is the final step in our progression of incremental wrapper/TAM design problems, and it includes $\mathcal{P}_{\mathbf{W}}$, $\mathcal{P}_{\mathbf{AW}}$ and $\mathcal{P}_{\mathbf{PAW}}$. An enumerative algorithm to solve $\mathcal{P}_{\mathbf{NPAW}}$ has been proposed, in which the search space can be pruned significantly when no further improvement to testing time would result.

We have applied our TAM optimization models to a realistic example SOC as well as to an industrial SOC; the experimental results demonstrate the feasibility of the proposed techniques. To the best of our knowledge, this is the first reported attempt at integrated wrapper/TAM co-optimization that has been applied to an industrial SOC.

In future work, we intend to extend our TAM optimization models to include several other TAM configurations, including daisy-chained cores on TAMs [1] and "forked and merged" TAMs [5]. We intend to extend our models, such that multiple wrappers on the same TAM are active in the test data transfer mode at the same time; this will allow us to address the problems of both testing hierarchical cores, as well as *Extest*. While ILP is a useful optimization tool for reasonably-sized problem instances, execution times can increase significantly for complex SOCs and large values of B. This is also true of our enumerative approach to Problems $\mathcal{P}_{\mathbf{PAW}}$ and $\mathcal{P}_{\mathbf{NPAW}}$. We are in the process of designing heuristic algorithms for each of the problems formulated in this paper that can efficiently address wrapper/TAM co-optimization for large TAM widths as well as large numbers of TAMs. Furthermore, we plan to add constraints related to power dissipation, routing complexity and layout area to our TAM optimization models.

Acknowledgments

The authors thank Harry van Herten and Erwin Waterlander for their help with providing data for the Philips SOC p93791, Henk Hollmann and Wil Schilders for their help with partition theory, and Jan Korst for his help with the $\mathcal{NP}$-hardness proof for $\mathcal{P}_{\mathbf{NPAW}}$. We also thank Sandeep Koranne and Harald Vranken for their constructive review comments on earlier versions of this paper.

References

1. J. Aerts and E.J. Marinissen, "Scan Chain Design for Test Time Reduction in Core-Based ICs," in *Proc. International Test Conference*, 1998, pp. 448–457.
2. M. Berkelaar, *lpsolve 3.0*, Eindhoven University of Technology, Eindhoven, The Netherlands. *ftp://ftp.ics.ele.tue.nl/pub/lp_solve*.
3. K. Chakrabarty, "Design of System-on-a-Chip Test Access Architectures Using Integer Linear Programming," in *Proc. VLSI Test Symposium*, 2000, pp. 127–134.
4. K. Chakrabarty, "Design of System-on-a-Chip Test Access Architectures Under Place-and-Route and Power Constraints," in *Proc. Design Automation Conference*, 2000, pp. 432–437.
5. K. Chakrabarty, "Optimal Test Access Architectures for System-on-a-Chip," *ACM Transactions on Design Automation of Electronic Systems*, vol. 6, pp. 26–49, Jan. 2001.
6. T.J. Chakraborty, S. Bhawmik, and C-.H. Chiang, "Test Access Methodology for System-on-Chip Testing," *Digest of the International Workshop on Testing Embedded Core-Based System-Chips*, 2000, pp. 1.1-1–1.1-7.
7. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San Francisco, CA: W.H. Freeman and Co., 1979.
8. I. Ghosh, S. Dey, and N.K. Jha, "A Fast and Low Cost Testing Technique for Core-Based System-on-Chip," in *Proc. Design Automation Conference*, 1998, pp. 542–547.
9. P. Harrod, "Testing Re-Usable IP: A Case Study," in *Proc. International Test Conference*, 1999, pp. 493–498.
10. IEEE P1500 Standard for Embedded Core Test. *http://grouper.ieee.org/groups/1500/*.
11. V. Immaneni and S. Raman, "Direct Access Test Scheme—Design of Block and Core Cells for Embedded ASICs," in *Proc. International Test Conference*, 1990, pp. 488–492.
12. E. Larsson and Z. Peng, "An Integrated System-on-Chip Test Framework," in *Proc. Design, Automation, and Test in Europe (DATE)*, 2001, pp. 138–144.
13. J.H. van Lint and R.M. Wilson, *A Course in Combinatorics*, Cambridge: Cambridge University Press, 1992.
14. E.J. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousbera, and C. Wouters, "A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores," in *Proc. International Test Conference*, 1998, pp. 284–293.
15. E.J. Marinissen, S.K. Goel, and M. Lousberg, "Wrapper Design for Embedded Core Test," in *Proc. International Test Conference*, 2000, pp. 911–920.
16. E.J. Marinissen, R. Kapur, and Y. Zorian, "On Using IEEE P1500 SECT for Test Plug-n-Play," in *Proc. International Test Conference*, 2000, pp. 770–777.
17. M. Nourani and C. Papachristou, "An ILP Formulation to Optimize Test Access Mechanism in System-on-Chip Testing," "*IEEE International Test Conference*, 2000, pp. 902–910.
18. Semiconductor Industry Association, *International Technology Roadmap for Semiconductors, http://public.itrs.net/files/1999_SIA_Roadmap/Home.htm*.
19. N.A. Touba and B. Pouya, "Using Partial Isolation Rings to Test Core-Based Designs," *IEEE Design and Test of Computers*, vol. 14, pp. 52–59, Oct.–Dec. 1997.
20. P. Varma and S. Bhatia, "A Structured Test Re-Use Methodology for Core-Based System Chips," in *Proc. International Test Conference*, 1998, pp. 294–302.

21. H.P. Williams, *Model Building in Mathematical Programming*. 2nd ed., New York, NY: John Wiley, 1985.
22. Y. Zorian, E.J. Marinissen, and S. Dey, "Testing Embedded-Core-Based System Chips," in *Proc. International Test Conference*, 1998, pp. 130–143.

Vikram Iyengar received the B.E. degree in Electrical and Electronics Engineering from the Birla Institute of Technology, India, in 1996, and the M.S. degree in Computer Engineering from Boston University in 1998. He is presently a Ph.D. candidate and research assistant in Computer Engineering at Duke University. Mr. Iyengar has published over 20 papers in archival journals, refereed conference proceedings, and workshops. He is currently working on research projects related to test scheduling and test delivery architectures for system-on-chip designs. Mr. Iyengar received an IBM Graduate Fellowship in 2001. Mr. Iyengar is a member of IEEE.

Krishnendu Chakrabarty received the B.Tech. degree from the Indian Institute of Technology, Kharagpur, in 1990, and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1992 and 1995, respectively, all in Computer Science and Engineering. He is now Assistant Professor of Electrical and Computer Engineering at Duke University. Dr. Chakrabarty is a recipient of the National Science Foundation Early Faculty (CAREER) award, the Office of Naval Research Young Investigator award and the Mercator Professor award from Deutsche Forschungsgemeinschaft, Germany. His current research projects (supported by NSF, DARPA, ONR, and industrial sponsors) are in system-on-chip test, real-time operating systems, distributed sensor networks, and architectural optimization of microelectrofluidic systems. He has published over 80 papers in archival journals and refereed conference proceedings, and he holds a US patent in built-in self-test. He is a Senior Member of IEEE, member of ACM and ACM SIGDA, and member of Sigma Xi. He serves as an Associate Editor for IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, and as an Editor for the Journal of Electronic Testing: Theory and Applications. He also serves as Vice Chair of Technical Activities in IEEE's Test Technology Technical Council, and is a member of the program committees of several IEEE/ACM conferences and workshops.

Erik Jan Marinissen is Senior Member of Scientific Staff at Philips Research Laboratories in Eindhoven, The Netherlands. He holds an M.Sc. degree in Computing Science (1990) and an MTD (Master of Technological Design) degree in Software Technology (1992), both from Eindhoven University of Technology. Marinissen's research interests include all topics in the domain of test and debug of digital VLSI circuits. He has published over 45 journal and conference papers, holds one US patent, and has several US and EP patents pending in the domain of core test and other digital test fields. He is the recipient of the Best Paper Award of the Chrysler-Delco-Ford Automotive Electronics Reliability Workshop 1995. Marinissen is a Senior Member of IEEE, and member of VIE, XOOTIC, and Philips CTAG. He serves as Editor-in-Chief of the IEEE P1500 Standard for Embedded Core Test and is a member of the organizing and program committees of DATE, DDECS, ETW, ITSW, LATW, SBCCI, and TECS. He has presented numerous tutorials on core-based testing within Philips and at international conferences. Marinissen serves as Member of the Editorial Board of JETTA.