Design and Optimization of Multi-level TAM Architectures for Hierarchical SOC¹

Vikram Iyengar[†], Krishnendu Chakrabarty[‡], Mark D. Krasniewski[‡] and Gopind N. Kumar[‡]

[†]IBM Microelectronics

Essex Jct, VT 05452

vikrami@us.ibm.com

[‡]Electrical & Computer Engineering, Duke University

Durham, NC 27708

{krish,mdk3,gmk2}@ee.duke.edu

Abstract

Multi-level TAM optimization is necessary for modular testing of hierarchical SOC's that contain older-generation SOC's as embedded cores. We present two hierarchical TAM optimization flows that exploit recent advances in TAM design for flattened SOC hierarchies. Unlike prior methods that assume flat test hierarchies, the proposed methods are directly applicable to real-world design transfer models between the core vendor and the SOC integrator. Experimental results are presented for four ITC'02 SOC Test Benchmarks.

1 Introduction

A hierarchical system-on-chip (SOC) is designed by integrating heterogeneous technology cores at several layers of hierarchy. The ability to re-use embedded functionality has led to the paradigm of "today's SOC is tomorrow's embedded core" [1]. Two broad design transfer models are emerging in hierarchical SOC design flows.

1. Non-interactive: In this model, there is limited communication between the core vendor and the SOC integrator. The cores are taken off-the-shelf and integrated into designs "as is".

2. Interactive: Here, there is a certain amount of communication between the core vendor and core user during system integration.

While hierarchical SOC's offer reduced cost and rapid system implementation, they pose several difficult test challenges. *Modular testing* of the embedded cores in an SOC can simplify the complex problems of test access and application [2]. For modular testing, an embedded core is isolated from surrounding logic using a test wrapper, and a test access mechanism (TAM) is designed to deliver test data from the I/O pins of the SOC. This facilitates the reuse of precomputed tests for individual cores and partitions the SOC for test; thus the test methodology follows the modular design process.

The problem of multi-level TAM design and optimization for hierarchical SOC's has not been systematically addressed in the literature. In most prior work on TAM design, the SOC hierarchy is assumed to be flattened for the purpose of test [3, 4, 5, 6, 7]. However, this assumption is often unrealistic in practice, especially when older-generation SOC's are themselves used as embedded cores in new SOC designs. In such cases, the core vendor may have already designed a TAM within the "mega-core" that is provided to the SOC integrator. A *mega-core* is defined as a design that contains non-mergeable embedded cores. In order to ensure effective testing of an SOC based on mega-cores, the

top-level TAM must communicate with lower level TAMs within mega-cores. The top-level TAMs must also be wide enough to fork out to the pre-designed lower-level TAMs. Moreover, the system-level test architecture must be able to reuse the existing test architecture within cores; redesign of core test structures must be kept to a minimum and must be consistent with the design transfer model between the core designer and the core user [8].

Figure 1 illustrates a hierarchical SOC (referred to as Module 0 in [9]) having two top-level cores A and B, and one top-level mega-core C. Core C contains Cores D and H, and a lower-level mega-core E, which in turn contains Cores F and G. A Level-1 TAM is connected to Cores A, B, and C. This TAM connects to a Level-2 TAM within Core C for the testing of Cores C, D, H, and E. The Level-2 TAM connects to a Level-3 TAM within Core E. The Level-3 TAM is used to test Cores F and G.

Three proposals for test access to hierarchical embedded cores were recently presented in [10, 11, 12]. In [10], the design of a test bus architecture based on scan switches was discussed. However, hierarchical TAMs that transport test data to top-level cores and also to lower-level TAMs within mega-cores were not considered. In [11], the design of a hierarchical TAM was described; however the lower-level TAMs were limited to 1-bit boundary scan chains; multi-level test buses were not considered. In [12], the implementation of a hierarchical test bus architecture was described. However, no attempt was made to optimize these multi-level TAMs to minimize testing time. In particular, none of the three proposals considered the optimization of multi-level TAMs for cores embedded within other cores.

A TAM design methodology that closely follows the design transfer model in use is critical because if the core vendor has implemented "hard" (i.e., non-alterable) TAMs within mega-cores, the SOC integrator must take into account these lower-level TAM widths while optimizing the widths and core assignment for higher-level TAMs. On the other hand, if the core vendor designs TAMs within mega-cores in consultation with the SOC integrator, the system designer's TAM optimization method must be flexible enough to include parameters for lower-level cores. Finally, multi-level TAM design for SOC's that include reused cores at multiple levels is needed to exploit "TAM reuse" and "wrapper reuse" in the test development process.

In this paper, we describe the optimization of multi-level TAMs for the "cores within cores" design paradigm. We do not present new algorithms for TAM optimization here; instead, we show how known methods for flattened SOC's can be used for multi-level TAM optimization in hierarchical SOC's. TAM widths are calculated for higher- and lower-level TAMs using a combination of integer linear programming (ILP) and enumeration [5], and efficient heuristics [13]. The methods presented here, unlike prior methods

¹This research was supported in part by the National Science Foundation under grants CCR-9875324 and CCR-0204077.

Figure 1. Illustration of a multi-level TAM architecture for a hierarchical SOC, referred to as Module 0 in [9].

that assume flat test hierarchies, are therefore directly applicable to test development for SOC in real-world design transfer models. We present experimental results for four hierarchical benchmark SOC [9] to demonstrate the effectiveness of the proposed method. In this work, we do not consider the top-level tests for interconnects and user-defined logic. The proposed design approach can be easily extended to handle these tests.

2 Review of TAM optimization methods

In this section, we review TAM optimization methods based on ILP and enumeration, and efficient heuristics. A multi-level TAM design method based on extensions of these “flat TAM” design methods will later be presented in Section 3.

2.1 ILP and enumeration

In [5], the total TAM width is partitioned among a number of test buses and each core is assigned to one of these TAMs. Given an SOC having N cores, the optimization problem in [5] is formulated as follows.

Determine (i) the number B of TAMs for the SOC, (ii) a partition of the total TAM width W among the TAMs, (iii) an assignment of the N cores to TAMs, and (iv) a wrapper design for each core, such that SOC testing time is minimized.

The problem of wrapper design was solved using the *Design-wrapper* algorithm [5] based on the Best Fit Decreasing heuristic. The core assignment problem was solved using an ILP model constructed as follows.

Consider an SOC consisting of N cores and B TAMs of widths $w_1, w_2, \dots, w_B$. The time taken to test Core i assigned to TAM j is denoted by $T_i(w_j)$ clock cycles. Let x_{ij} be a binary variable, which is defined to be 1, if Core i is assigned to TAM j , and 0 otherwise.

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 $\max_{1 \leq j \leq B} \sum_{i=1}^N T_i(w_j) \cdot x_{ij}$. The ILP model for core assignment can be formulated as follows.

Objective: Minimize testing time $\mathcal{T}$, 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

In [5], it was observed that the execution time of the above ILP model was very small. This was exploited to determine the number

of TAMs and their widths for the SOC by selectively enumerating width partitions using a bounding function and selecting the best partition.

2.2 Heuristics for TAM width optimization

While optimal results were obtained in [5], the number of TAMs designed was small in order to maintain feasible compute times. However, if a larger number of TAMs is designed, the testing time can often be reduced. This is because when there are multiple TAMs of different widths, cores have a greater chance of being assigned to a TAM whose width matches the cores' own test data requirements; thus the number of unnecessary (idle) TAM wires assigned to cores is reduced. Moreover, multiple TAMs provide greater test parallelism.

In [13], a three-step heuristic method was presented to design TAM architectures for large SOC containing multiple TAMs. In the first step, a heuristic algorithm *Core_assign* was used for core assignment to TAMs. In the second step, *Core_assign* was used to rapidly enumerate and evaluate width partitions for a large number of TAMs. The partition enumeration algorithm employs several levels of solution-space pruning during width partition evaluation. The number of partitions enumerated is also significantly limited by establishing upper bounds on each TAM width variable w_j during enumeration [13]. This provides a fast approximation for the optimal values of TAM width partition and testing time. The testing time was reduced further in [13] by a third optimization step based on a one-time use of the ILP model for core assignment.

3 Multi-level TAM optimization

In this section, we describe how the methods reviewed in Section 2 can be applied to optimize multi-level TAMs for hierarchical SOC. The top-level SOC is composed of embedded cores as well as embedded mega-cores obtained from core vendors. A mega-core may have been a stand-alone SOC in an earlier generation or is a complex circuit requiring the core vendor to instantiate off-the-shelf cores in its design.

Mega-cores may be supplied by core vendors in varying degrees of readiness for test integration. For example, the IEEE P1500 proposal on embedded core test defines two compliance levels for core delivery: 1500-wrapped and 1500-unwrapped [15]. Here we describe three other scenarios, based in part on the P1500 compliance levels. We use the term *wrapped* to denote a core for which a wrapper has been pre-designed, as in [15]. We use the term *TAM-ed* to denote a mega-core that contains an internal TAM structure.

1. Not TAM-ed and not wrapped: In this scenario, the system integrator must design a wrapper for the mega-core as well as TAMs within the mega-core. The mega-cores are therefore delivered either as soft-cores or before final netlist and layout optimization, such that TAMs can be inserted within the mega-cores.
2. TAM-ed but not wrapped: In this scenario, the mega-core contains lower-level TAMs, however a wrapper for it is still required to be designed by the core integrator. Knowledge of the number and lengths of top-level scan chains as well as testing times of lower-level cores are therefore required by the system integrator to design balanced top-level wrapper scan chains for the mega-core.
3. TAM-ed and wrapped: In this scenario, we consider TAM-ed mega-cores for which wrappers have been designed by the core

vendor. This scenario is especially suitable for a mega-core that was an SOC in an earlier generation. The width of the TAM that must be supplied to it is pre-specified. The I/O and scan chain terminals for wrapped hard-IP sub-cores may not be available to the SOC integrator to perform further test width adaptation. Such a mega-core may also contain its own test control block (e.g., Core C in Figure 1) with strict test protocols specified. Therefore, we assume that such mega-cores are wrapped by the core vendors prior to design transfer and test data cannot be further serialized or parallelized by the SOC integrator. At system level, only structures that facilitate normal/test operation, interconnect test, and bypass are created.

We address the problem of TAM optimization for multi-level SOC in a hierarchical manner. Our work is targeted at TAM optimization for SOC containing hard-cores that represent critical-IP. We therefore address Scenario 3 in this work. In Scenario 3, the mega-cores are wrapped and TAM-ed by the core-vendor, either in the non-interactive or interactive design transfer model.

3.1 Non-interactive design transfer model

In the non-interactive design transfer model, the core vendor designs and implements TAM architectures for use within the mega-cores. Width optimization for these lower-level TAMs is performed without input from the SOC integrator, and testing times for mega-cores are specified prior to design transfer. The test parameters supplied by the core vendor to the SOC integrator for non-hierarchical cores include the number of primary (including bidirectional) I/Os, test patterns, scan chains, and scan chain lengths. The parameters supplied for the mega-cores include only the TAM width and testing time.

The multi-level TAM optimization problem in the non-interactive design transfer model can now be stated as follows.

$\mathcal{P}_{\text{non-int}}$: Given the test set parameters for the top-level cores and the total TAM width W for the SOC, determine a wrapper design for each core, and a partition of W among the cores in the test schedule, such that the SOC testing time is minimized under the constraints that (i) W is not exceeded at any time, (ii) the mega-cores receive at least their pre-specified TAM widths, and (iii) parent mega-cores are tested only after their embedded child cores.

A special case of $\mathcal{P}_{\text{non-int}}$ that contains no mega-cores is equivalent to $\mathcal{P}_{\text{co-opt}}$ in [5]; $\mathcal{P}_{\text{non-int}}$ is therefore $\mathcal{NP}$ -hard.

There are two reasons for incorporating Constraint (iii) in the problem statement. Firstly, memories are often embedded within logic cores, e.g., the Philips cores presented in [9]. These embedded memories must often be tested and repaired before the logic surrounding them is tested because the memories may be used during logic core test. Secondly, the test for large mega-cores can be halted as soon as a smaller embedded component fails.

Figure 2 presents the TAM design flow for hierarchical SOC in the non-interactive design transfer model. Figure 2 is not the pseudocode for a TAM optimization algorithm, but it illustrates the overall TAM design flow from core vendor to system integrator. Lines 1 through 5 represent the TAM design flow for the mega-cores (performed by the core vendor). Note from the “bottom-up” guideline in Line 1 that the TAM optimization methodology presented here is hierarchical. For example, in Figure 1, the Level-3 TAM in Core E is optimized before the Level-2 TAM in Core C. Moreover, TAM optimization for mega-cores containing lower-

Design Flow *Non-interactive_Hierarchical_TAM_Optimizer()*

Mega-core vendor:

- 1 For each Mega-core i (starting bottom-up in the hierarchy)
- /* Let W_i be the total TAM width for i */
- 2 Partition W_i among the embedded cores in i and determine the test schedule using an approach from [5, 13];
- 3 Schedule the test for the interconnect logic within i
- 4 Determine the total testing time $\mathcal{T}_i$ for Mega-core i ;
- 5 Implement TAM architecture for i prior to design transfer;

SOC integrator:

- /* Let W be the total TAM width for the SOC */
 - 6 Obtain the core test parameters for the top-level non-hierarchical cores;
 - 7 For each top-level Mega-core i in the SOC
 - 8 Obtain the testing time $\mathcal{T}_i$ and specified TAM width W_i ;
 - /* Let $\mathcal{T}_i(w_j)$ be the SOC-level testing time of Core i on TAM width w_j */
 - 9 Set $\mathcal{T}_i(w_j) = \mathcal{T}_i$, for $w_j \geq W_i$
 - 10 Set $\mathcal{T}_i(w_j) = \infty$, for $w_j < W_i$
 - 11 Partition W among the cores using a TAM optimization technique, e.g. from [5, 13], and determine the test schedule;
 - 12 Implement system-level TAM architecture.
-

Figure 2. TAM design flow for non-interactive design transfer.

level mega-cores must also follow the design transfer model in use by the lower-level mega-core vendor. The total TAM width W_i for each Mega-core i in Line 2 is determined by the core vendor either from the number of test pins available or from the existing TAM architecture (if i was a TAM-ed SOC in a previous generation). In Lines 2 to 4, the TAM design and total testing time $\mathcal{T}_i$ for Mega-core i is determined.

Lines 6 through 12 present the SOC-level TAM design flow (performed by the system integrator). Core test parameters are obtained from the core vendors in Lines 6 and 8. In Lines 9 and 10, the TAM width assigned to each Mega-core i is “hardwired” to W_i by setting its testing time $\mathcal{T}_i(w)$ to $\mathcal{T}_i$ for $w_j \geq W_i$, and to ∞ for $w_j < W_i$. The result of this TAM width assignment for the TAM design methods [5, 13] used in Line 11 is as follows. The set of $\mathcal{T}_i(w_j)$ testing time–TAM width variables for each Mega-core i is reduced to the single constant $\mathcal{T}_i$ for $w_j \geq W_i$, prior to TAM optimization. If $w_j < W_i$, $\mathcal{T}_i(w_j)$ is set to ∞ . In Line 11, top-level TAM optimization is carried out for the SOC using a method from [5, 13] to obtain the final testing time for total TAM width W . The system-level TAM architecture is implemented in Line 12.

3.2 Interactive design transfer model

In the interactive design transfer model, the core vendor once again designs and implements TAM architectures for use within the mega-cores. However, the system integrator is now able to influence the choice of TAM width supplied to mega-cores by the core vendors based upon system-level TAM width requirements of other cores. The test parameters for each Mega-core i supplied by the core vendor to the SOC integrator prior to system-level TAM design therefore include a set of 2-tuples $\{(W_i, \mathcal{T}_i)\}$, where each tuple represents a potential TAM width–testing time choice for the mega-core, and the number of tuples depends on the guidelines from the core user to the core vendor.

The multi-level TAM optimization problem in the interactive design transfer model can now be stated as follows.

Design Flow *Interactive_Hierarchical_TAM_Optimizer()*

Mega-core vendor:

```
1 For each Mega-core  $i$  (starting bottom-up in the hierarchy)
/* Let  $W_{i,max}$  be the maximum allowable total TAM width for  $i$  */
2 For each  $W_i$  from 1 to  $W_{i,max}$ 
3 Partition  $W_i$  among the embedded cores in  $i$  and determine the test
  schedule using an approach from [5, 13];
4 Schedule the test for the interconnect logic within  $i$ ;
5 Determine the total testing time  $T_i(W_i)$ ;
```

SOC integrator:

```
/* Let  $W$  be the total TAM width for the SOC */
6 Obtain the core test parameters for the top-level non-hierarchical cores;
7 For each top-level Mega-core  $i$  in the SOC
8 Obtain the set of values  $\{(W_i, T_i)\}$ ;
/* Let  $T_i(w_j)$  be the SOC-level testing time of Core  $i$  on TAM width  $w_j$  */
9 For each value of  $W_i$ 
10 Set  $T_i(w_j) = T_i(W_i)$ , for  $w_j = W_i$ ;
11 Set  $T_i(w_j) = \infty$ , for  $w_j \neq W_i$ ;
12 Partition  $W$  among the cores using a TAM optimization approach
  from [5, 13] and determine the test schedule;
13 Communicate desired TAM widths for mega-cores to core vendors;
```

Mega-core vendor:

```
14 Implement TAM architectures for mega-cores prior to design transfer;
```

SOC integrator:

```
15 Implement system-level TAM architecture.
```

Figure 3. TAM design flow for interactive design transfer.

$\mathcal{P}_{int}$: Given the test set parameters for the top-level cores and W for the SOC, determine a wrapper design for each core, and a partition of W among the cores in the test schedule, such that the SOC testing time is minimized under the constraints that (i) W is not exceeded at any time, (ii) each mega-core receives *one* of its pre-specified TAM widths, and (iii) parent mega-cores are tested only after their embedded child cores.

A special case of $\mathcal{P}_{int}$ in which each mega-core has only one pre-specified TAM width is equivalent to $\mathcal{P}_{non-int}$ in Section 3.1; $\mathcal{P}_{int}$ is therefore $\mathcal{NP}$ -hard.

Figure 3 presents the TAM design flow for the interactive design transfer model. Lines 1 through 5 represent the TAM design flow for the mega-cores (performed by the core vendor). TAM design is now explored for an entire range of width values to estimate potential testing times. While this takes more computation time than a single TAM design calculation, it results in lower SOC testing times as will be seen in Section 4. The value of $W_{i,max}$ is chosen by the core vendor depending on test pin, layout and overhead constraints for Mega-core i . Lines 6 through 12 represent the SOC-level TAM design flow (performed by the system integrator). In Lines 10 and 11, the set of $T_i(w_j)$ variables for each Mega-core i is reduced to the set of $T_i(W_i)$ values. In Lines 12 and 13, the SOC-level TAM architecture is designed and desired TAM widths for mega-cores are communicated to core vendors. The core vendors then implement the mega-core TAM architectures and transfer their designs to the SOC integrator. The system-level TAM architecture is implemented in Line 15.

4 Case studies

In this section, we present two case studies performed using the proposed methodology. We present experimental results for four SOC: p22810, p34392, p93791, and a586710 from the *ITC'02*

SOC test benchmarks [9]. The experimental results were obtained using a Sun Ultra 10 workstation with a 333 MHz processor and 256 MB memory.

4.1 Non-interactive design transfer model

We first performed TAM optimization using the design flow in Figure 2. In Table 1, we compare the testing times (in clock cycles) and CPU times (in seconds) of the proposed hierarchical TAM optimization method with those of the corresponding “flat” methods in [5, 13]. Though hierarchical TAM optimization was performed based on both methods in [5, 13] for each SOC, results are presented here for only one of the two methods for each SOC due to insufficient space. The testing times for the “flat” and hierarchical methods are denoted by $\mathcal{T}_{flat}$ and $\mathcal{T}_{hier}$, respectively. Similarly, the CPU times for the “flat” and hierarchical methods are denoted by $\mathcal{E}_{flat}$ and $\mathcal{E}_{hier}$, respectively. The percentage change in testing time $\Delta\mathcal{T}$ using hierarchical TAM optimization is calculated as $\frac{\mathcal{T}_{hier} - \mathcal{T}_{flat}}{\mathcal{T}_{flat}} \times 100$.

In Table 1(a), we present results for p22810. The hierarchical TAM optimization flow was based on the ILP and enumeration method of [5]. TAM widths supplied to each mega-core were fixed at 8 bits prior to system-level TAM design. While the testing times obtained are higher than those obtained by (unrealistically) assuming that the SOC hierarchy can be flattened, the results obtained here are more realistic for hierarchical TAM design. Note from Table 1(a) that the testing time for the hierarchical method levels out at 366260 cycles at $W = 40$. A total top-level TAM width of 40 is therefore an effective choice for this SOC.

In Table 1(b), we present results for p34392. Hierarchical TAM optimization was performed based on the heuristic method of [13]. A TAM width of 16 was supplied to each mega-core prior to system-level TAM optimization. The increase in testing time over the “flat” case is between 13.59% and 84.56%. The testing time for the flat method reaches a minimum value of 544579 cycles at $W = 40$. This lower bound (observed earlier in [5, 7, 13]) is the lower bound on testing time for Core 18 (the bottleneck core) in the system. In the hierarchical method, since Core 18 contains Core 19, its test is scheduled only after that for Core 19 completes. The combined testing times for Cores 18 and 19 is 618597 cycles; this is therefore the lower bound for the SOC using hierarchical TAM optimization. The CPU times for the hierarchical method are significantly lower than those for the flat method. This is because in the flat method, an ILP model is run as a final step for all 19 cores of p34392 simultaneously, thereby taking a longer CPU time. In the hierarchical optimization flow, the SOC is partitioned per mega-core and the ILP model runs significantly faster.

In Table 1(c), we present results for p93791. Hierarchical TAM optimization was performed using the heuristic method [13]. A TAM width of 16 was supplied to each mega-core prior to system-level TAM optimization. In Table 1(d), we present results for a586710. Hierarchical TAM optimization was performed using ILP and enumeration [5]. A TAM width of 8 was supplied to each mega-core prior to system-level TAM optimization.

4.2 Interactive design transfer model

We next carried out TAM optimization using the design flow in Figure 3. A set of potential TAM widths and corresponding testing times for each mega-core was calculated prior to system-level

W	Method in [5]		Hierarchical method		$\Delta \mathcal{T}$ (%)	$\frac{\mathcal{E}_{hier}}{\mathcal{E}_{flat}}$
	$\mathcal{T}_{flat}$	$\mathcal{E}_{flat}$	$\mathcal{T}_{hier}$	$\mathcal{E}_{hier}$		
16	462210	11	505858	31	9.44	2.81
24	361571	24	412682	92	14.14	3.83
32	312659	49	396473	83	26.81	1.69
40	278359	60	366260	77	31.58	1.28
48	268472	84	366260	116	36.42	1.38
64	260638	122	366260	115	40.52	0.94

(a)

W	Method in [13]		Hierarchical method		$\Delta \mathcal{T}$ (%)	$\frac{\mathcal{E}_{hier}}{\mathcal{E}_{flat}}$
	$\mathcal{T}_{flat}$	$\mathcal{E}_{flat}$	$\mathcal{T}_{hier}$	$\mathcal{E}_{hier}$		
24	729872	27	1347023	1	84.56	0.04
32	663193	51	788873	1	18.95	0.02
40	544579	39	728426	2	33.76	0.05
48	544579	205	618597	7	13.59	0.03
56	544579	109	618597	8	13.59	0.07
64	544579	288	618597	9	13.59	0.03

(b)

W	Method in [13]		Hierarchical method		$\Delta \mathcal{T}$ (%)	$\frac{\mathcal{E}_{hier}}{\mathcal{E}_{flat}}$
	$\mathcal{T}_{flat}$	$\mathcal{E}_{flat}$	$\mathcal{T}_{hier}$	$\mathcal{E}_{hier}$		
16	1786200	2	2044124	5	14.44	2.50
24	1209420	3	1351710	6	11.77	2.00
32	894342	1	1087300	7	21.58	7.00
40	741965	1	839796	7	13.19	7.00
48	599373	3	839796	7	40.11	2.33
64	473997	2	839796	8	77.17	4.00

(c)

W	Method in [5]		Hierarchical method		$\Delta \mathcal{T}$ (%)	$\frac{\mathcal{E}_{hier}}{\mathcal{E}_{flat}}$
	$\mathcal{T}_{flat}$	$\mathcal{E}_{flat}$	$\mathcal{T}_{hier}$	$\mathcal{E}_{hier}$		
24	2.84×10^7	7	3.06×10^7	14	7.75	2.00
32	2.10×10^7	2	2.88×10^7	19	37.14	9.50
40	1.91×10^7	4	2.50×10^7	26	30.89	4.21
48	1.52×10^7	4	2.14×10^7	41	40.79	10.25
56	1.41×10^7	4	2.14×10^7	42	51.77	10.50
64	1.41×10^7	5	2.14×10^7	115	51.77	23.00

(d)

Table 1. Results for the non-interactive case for (a) p22810 using ILP and enumeration [5], (b) p34392 using the heuristic method of [13], (c) p93791 using the heuristic method, and (d) a586710 using ILP and enumeration.

TAM optimization. The best TAM width for each mega-core was identified at the system level and the final SOC testing time was then determined. In Table 2, we compare the testing times and CPU times for hierarchical TAM optimization with those for the corresponding flat methods in [5, 13]. We do not list the testing times and CPU times of the flat methods, since these are already listed in Table 1. We also compare the testing times for the interactive design transfer model with the testing times for the non-interactive model.

In Table 2(a), we present results for p22810. The hierarchical TAM optimization flow was based on the ILP and enumeration method of [5]. The testing times obtained are very close to those for flat TAM design, thereby demonstrating that multi-level TAM optimization for hierarchical SOC can indeed be performed with effective results. More importantly, testing times here are significantly lower than those for the non-interactive case; see last column of Table 2(a). This is a result of the greater flexibility in choosing lower-level TAM widths on the basis of system-level optimization.

In Table 2(b), we present results for p34392. Hierarchical TAM optimization was performed based on the heuristic method of [13]. The testing times are once again lower than those for the non-interactive case. In Figure 4, we illustrate the multi-level TAM architecture designed for p34392 in the interactive design trans-

W	Hierarchical method		Compared to [5]		Compared to non-interactive: $\Delta \mathcal{T}$ (%)
	$\mathcal{T}_{hier}$	$\mathcal{E}_{hier}$	$\Delta \mathcal{T}$ (%)	$\frac{\mathcal{E}_{hier}}{\mathcal{E}_{flat}}$	
16	483795	23	4.66	2.09	-4.36
24	363812	56	0.62	2.33	-11.84
32	324079	120	3.65	2.45	-18.26
40	288159	79	3.52	1.32	-21.32
48	272892	129	1.65	1.54	-25.49
64	263443	187	1.08	1.53	-28.07

(a)

W	Hierarchical method		Compared to [13]		Compared to non-interactive: $\Delta \mathcal{T}$ (%)
	$\mathcal{T}_{hier}$	$\mathcal{E}_{hier}$	$\Delta \mathcal{T}$ (%)	$\frac{\mathcal{E}_{hier}}{\mathcal{E}_{flat}}$	
24	819176	1	12.24	0.04	-39.18
32	713174	1	7.54	0.02	-9.60
40	606261	2	11.33	0.05	-16.77
48	593925	4	9.06	0.02	-3.98
56	581588	4	6.79	0.04	-5.98
64	581588	4	6.79	0.01	-5.98

(b)

W	Hierarchical method		Compared to [13]		Compared to non-interactive: $\Delta \mathcal{T}$ (%)
	$\mathcal{T}_{hier}$	$\mathcal{E}_{hier}$	$\Delta \mathcal{T}$ (%)	$\frac{\mathcal{E}_{hier}}{\mathcal{E}_{flat}}$	
16	1927010	8	7.88	4.00	-5.73
24	1293990	14	6.99	4.67	-4.27
32	863765	22	-3.42	22.00	-20.56
40	758156	37	2.18	37.00	-9.72
48	635710	68	6.06	22.67	-24.30
64	481896	174	1.67	87	-42.62

(c)

W	Hierarchical method		Compared to [5]		Compared to non-interactive: $\Delta \mathcal{T}$ (%)
	$\mathcal{T}_{hier}$	$\mathcal{E}_{hier}$	$\Delta \mathcal{T}$ (%)	$\frac{\mathcal{E}_{hier}}{\mathcal{E}_{flat}}$	
24	3.06×10^7	30	7.75	4.29	0.00
32	2.11×10^7	24	0.48	12.00	-26.74
40	1.91×10^7	40	0.00	10.00	-23.60
48	1.75×10^7	45	15.13	11.25	-18.22
56	1.70×10^7	46	20.57	11.50	-20.57
64	1.63×10^7	56	15.60	11.2	-23.83

(d)

Table 2. Results for the interactive case for (a) p22810 using ILP and enumeration [5], (b) p34392 using the heuristic method of [13], (c) p93791 using the heuristic method, and (d) a586710 using ILP and enumeration.

fer model. The flat TAM architecture, from [13], has five TAMs, while the hierarchical TAM architecture consists of three Level-1 TAMs and five Level-2 TAMs in simultaneous operation.

In Table 2(c), we present hierarchical TAM optimization results for p93791 performed using the heuristic method [13]. The increase in testing time over the flat case ranges from 1.67% to 7.88%, except for $W = 32$, for which a decrease in testing time of 3.42% over the flat case is observed. We attribute this to the fact that the method in [13] is heuristic and inefficient TAM width assignments can sometimes be made when the search-space is large. Finally, in Table 2(d), we present results for a586710. Hierarchical TAM optimization was performed using ILP and enumeration [5]. Testing times were equal or lower than those obtained for the non-interactive case.

We compare the final results of TAM optimization using the flat, hierarchical non-interactive, and hierarchical interactive design flows for p93791 in Figures 5 and 6. (Figures are not drawn to scale.) TAM optimization is performed for $W = 40$. In Figure 5(a), we present the hierarchical structure of the SOC and its component cores. In Figure 5(b), we illustrate the test schedule obtained using the flat TAM optimization method of [13]. The total testing time for p93791 using the flat TAM optimization method is 741965 cycles. In Figure 6(a), we illustrate the test schedule obtained using the hierarchical non-interactive design flow. The

Figure 4. Illustration of the (a) TAM architecture for p34392 assuming flattened SOC [13], and (b) multi-level TAM architecture for actual SOC hierarchy.

testing time has increased to 839796 cycles. However, the TAM optimization flow now more closely follows the real-world design transfer model; therefore, the results obtained are more realistic. Finally, in Figure 6(b), we illustrate the test schedule obtained using hierarchical interactive design flow. There is a decrease of 81640 cycles in the testing time over the non-interactive case.

5 Conclusion

We have shown how TAM optimization methods proposed for “flattened SOC” can be used to solve the more realistic problem of designing multi-level TAM architectures for hierarchical SOC. Two TAM optimization flows have been proposed that are directly applicable to real-world design transfer models used by core vendors and SOC integrators. Experimental results for benchmark SOC indicate that testing times using hierarchical TAMs are comparable to those achieved using flat methods.

References

- [1] V. Chickermane et al. A building block BIST methodology for SOC designs: A case study. *Proc. Int. Test Conf.*, pp. 111–120, 2001.
- [2] Y. Zorian, E.J. Marinissen and S. Dey. Testing embedded-core-based system chips. *IEEE Computer*, vol. 32, pp. 52–60, June 1999.
- [3] E.J. Marinissen et al. A structured and scalable mechanism for test access to embedded reusable cores. *Proc. Int. Test Conf.*, pp. 284–293, 1998.
- [4] E. Larsson and Z. Peng. Test scheduling and scan-chain division under power constraint. *Proc. Asian Test Symp.*, pp. 259–264, 2001.
- [5] 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.
- [6] Y. Huang et al. On concurrent test of core-based SOC design. *J. Electronic Testing: Theory and Applications*, vol. 18, pp. 401–414, Aug–Oct 2002.
- [7] S.K. Goel and E.J. Marinissen. Effective and efficient test architecture design for SOC. *Proc. Int. Test Conf.*, pp. 529–538, 2002.
- [8] I. Parulkar et al. A scalable, low cost design-for-test architecture for UltraSPARCTM chip multi-processors. *Proc. Int. Test Conf.*, pp. 726–735, 2002.

Figure 5. Illustration of (a) hierarchical structure of p93791, (b) test schedule obtained using [13].

Figure 6. Illustration of the (a) test schedule for p93791 for the hierarchical non-interactive design flow, (b) test schedule for the hierarchical interactive design flow.

- [9] E.J. Marinissen, V. Iyengar and K. Chakrabarty. A set of benchmarks for modular testing of SOC. *Proc. Int. Test Conf.*, pp. 519–528, 2002. (<http://www.extra.research.philips.com/itc02socbenchm>).
- [10] T.J. Chakraborty, S. Bhawmik and C.-H. Chiang. Test access methodology for system-on-chip testing. *Proc. Int. TECS Workshop*, pp. 1.1–1.1-7, 2000.
- [11] M. Benabdenbi, W. Maroufi and M. Marzouki. CAS-BUS: A test access mechanism and a toolbox environment for core-based system chip testing. *JETTA*, vol. 18, pp. 455–473, Aug. 2002.
- [12] A. Benso et al. HD2BIST: A hierarchical framework for BIST scheduling, data patterns delivering and diagnosis in SOC. *Proc. Int. Test Conf.*, pp. 892–901, 2000.
- [13] V. Iyengar, K. Chakrabarty and E.J. Marinissen. Efficient wrapper/TAM co-optimization for large SOC. *Proc. Design Automation and Test in Europe Conf.*, pp. 491–498, 2002.
- [14] S. Koranne. On test scheduling for core-based SOC. *Proc. Int. Conf. VLSI Design*, pp. 505–510, 2002.
- [15] E.J. Marinissen et al. On IEEE P1500’s standard for embedded core test. *J. Electronic Testing: Theory and Applications*, vol. 18, pp. 365–383, Aug. 2002.