Multifrequency TAM Design for Hierarchical SOCs

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

***Abstract*—The emergence of megacores in hierarchical system-on-a-chip (SOC) presents new challenges to electronic test automation. This paper describes a new framework for designing test access mechanisms (TAMs) for modular testing of hierarchical SOCs. We first explore the concept that TAMs on the same level of design hierarchy employ multiple frequencies for test data transportation. Then we extend this concept to hierarchical SOCs and, by introducing frequency converters at the inputs and outputs of the megacores, the proposed solution not only removes the constraint that the system level TAM width must be wider than the internal TAM width of the megacores, but also facilitates rapid exploration of the tradeoffs between the test application time and the required test area. Experimental results for the ITC'02 SOC Test Benchmarks show that the proposed TAM design algorithms increase the size of the solution space that is explored, which, in turn, will lower the test application time when compared to the existing solutions.**

***Index Terms*—Electronic test, mega-cores, system-on-a-chip, test access mechanism.**

I. Introduction

To MEET stringent time-to-market requirements, modern system-on-a-chip (SOC) development is based on the design reuse philosophy, where two parties are involved: 1) core providers; and 2) system integrators. Core providers create libraries of predesigned and preverified building blocks, such as embedded microprocessors, network controllers, memories, and peripherals, known as embedded cores. System integrators put the SOC together by combining the available cores and their custom user-defined logic (UDL). Based on the very same design reuse philosophy, a new SOC design may contain old-generation SOCs as its embedded cores. In addition, some complex cores are composed of several smaller in-house/external cores due to functional requirements [6]. In this paper, we refer to such hierarchical cores in SOC designs as "megacores" and a lower-level core embedded in them as their "child core." Moreover, with the capability to integrate tens or even hundreds of millions of transistors onto a silicon die [13], SOC designs are becoming too large and complex to have only one level of hierarchy (SOC and cores), since the capabilities of electronic design automation tools and computing resources improve at a slower pace. As a result, state-of-the-art SOC designs often have multiple levels of hierarchy [5], [7], [30], [32].

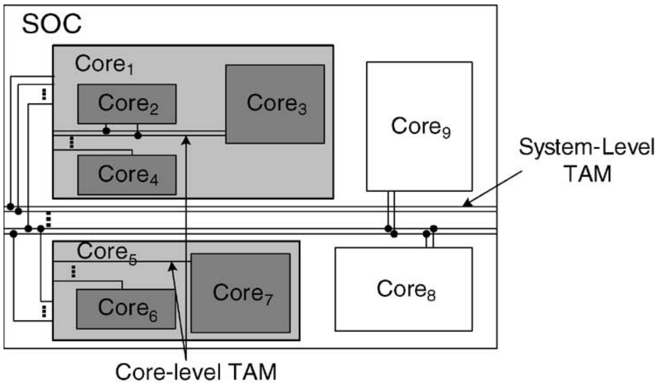


Fig. 1. Hierarchical SOC example.

While the reuse of intellectual property (IP) cores reduces the design cycle for SOC, the rapid increase in the transistor-to-pin ratio makes the manufacturing test development a major implementation bottleneck [39]. To address this bottleneck without affecting the design quality, test data must also be reused in the same modular fashion as the core functionality. To achieve this test reusability, an embedded core needs to be isolated during test using a core wrapper and a test access mechanism (TAM) is used to deliver test data from/to test sources/sinks [39]. The TAM design problem has been proven to be nondeterministic polynomial time (NP)-hard by Chakrabarty [4] and numerous TAM design algorithms have been proposed for optimizing the test application time. However, despite the recent research in TAM design, most of the prior work assumes that the hierarchy of the SOC is flattened during test, and hence, only a single-level TAM is needed. This assumption is becoming unrealistic because, as illustrated in Fig. 1, the hierarchical megacores may have the TAM already hardwired inside. In addition, a straightforward extension of the known TAM design algorithms will not necessarily lead to effective solutions for testing hierarchical SOCs. Moreover, most of the prior research on TAM design and test scheduling assumes that the automatic test equipment (ATE) operates at the same frequency (f_{ext}) as the embbeded cores' internal scan chain frequencies (f_{tam}). This assumption has been shown to be inefficient when there is a mismatch between the ATE capability and the internal scan chain speed [21]. Multifrequency TAM design, in which both the TAM width and the TAM running frequency assigned to each core are cooptimized, can facilitate SOC test cost reduction. Based on the above observations, the main objective of this paper is to investigate how we can use multiple frequencies for test data transportation in order to develop new TAM design algorithms for modular SOC testing, with special emphasis on hierarchical SOCs containing megacores.

Manuscript received September 7, 2004; revised December 21, 2004. Preliminary versions of this paper are published in the Proceedings of IEEE International Test Conference (ITC), pp. 1196–1202, October 2004, and in the Proceedings of the IEEE Asian Test Symposium (ATS), pp. 2–7, November 2004. This paper was recommended by Associate Editor K. Chakrabarty.

Q. Xu is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, New Territories, Hong Kong (e-mail: qxu@cse.cuhk.edu.hk).

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

Digital Object Identifier 10.1109/TCAD.2005.852440

The organization of this paper is as follows. Section II reviews the related approaches and outlines the main contributions of our work. Section III shows how multiple frequency test data transportation can be used for flattened SOCs, which is then extended to hierarchical SOCs in Section IV. Section V gives the experimental results and Section VI concludes the paper.

II. Related Work and Summary of Contributions

In this section, we first provide a comprehensive review of the related work on test architectures and their associated optimization algorithms. Once the known art is presented, we stress the distinguishing features of the approach presented in this paper.

A. Related Work on Single-Level TAM Design and Optimization

Aerts and Marinissen introduced three basic types of scalable test architectures in [1] called *Multiplexing*, *Daisychain*, and *Distribution* architectures. The common feature of the Multiplexing and Daisychain architectures is that all the cores get full access to the TAM lines, which is not the case for the Distribution architecture, where the TAM lines are distributed over all the cores. Two modular architectures, which can improve the test application time of the Multiplexing, Daisychain, and Distribution architectures, have been proposed in [27] and [37]: the *Test Bus* architecture [37] can be seen as a combination of the Multiplexing and the Distribution architectures, while the *TestRail* architecture [27] is a combination of the Daisychain and the Distribution architectures. Based on the TAM lines assignment strategy, the modular test architectures can be further categorized into two types [18].

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

Iyengar *et al.* [19] advocated the flexible-width architecture to be more effective because it improves the TAM line utilization. Since the test application time varies with TAM width as a "staircase" function for hard cores [15], only a few TAM widths (called *Pareto-optimal* points) between 1 and $W_{\max}$, the maximum number of TAM width, are relevant when assigning TAM resources to hard cores. The cores are frequently not assigned with a Pareto-optimal TAM width for fixed-width architecture, which wastes part of the test resources. Another reason is that test scheduling is usually more tightly integrated with TAM design in the flexible-width architecture, while in the fixed-width architecture, it is often performed after TAM design by shifting tests back and forth to satisfy the given constraints. However, due to the complexity of the SOC test problem, it cannot be generalized that the flexible-width approach is better than the fixed-width one. This is because of the following three reasons. 1) The Pareto-optimal TAM width only exists in hard cores for which internal scan chain design is fixed. For soft or firm cores, there is no waste of test resources when assigning them to any fixed-width TAM bus. 2) Because of the strong NP-hard attribute of the test architecture optimization problem, the size of the solution space for this problem is enormous, even with the fixed-width TAM constraint. Hence, the effectiveness of an approach is to a large extent dependent on the effectiveness of the search algorithm embedded in the approach instead of the architecture. 3) Because cores on the same TAM bus can share the same scan enable signal in fixed-width architecture, the test control pin requirement is usually lower for the fixed-width architecture than for the flexible-width architecture. These test control pins will reduce the available TAM width for test data transfer [9], which will impact significantly the SOCs with a large number of cores. Due to the nature of our approach, in this paper, we consider a fixed-width architecture. However, because we compare our results against both the flexible-width and fixed-width architectures, in the following, we provide a summary of the relevant techniques for both approaches.

Various optimization algorithms for fixed-width test bus architecture have been proposed. Iyengar *et al.* [15] first formulated the integrated wrapper/TAM cooptimization problem and broke it down into a progression of four incremental problems in the order of increasing complexity. An integer linear programming (ILP) model was then presented to solve the problem. To decrease the computation time of the ILP method, the same authors proposed a heuristic algorithm that prunes the solution space and reduces the computation time at the cost of increased test application time [16]. Instead of optimizing the finish time of the last core test, Koranne [24] proposed to optimize the average completion time of all the TAM partitions and solved the problem via reduction to the minimum weight perfect bipartite graph matching. By modeling the test sources/sinks as sources/sinks in a network, TAMs as channels with capacity proportional to their width, and each core under test as a node in the network, Koranne [23] formulated test scheduling as a network transportation problem and presented a 2-approximation algorithm to solve it by using the results of single source unsplittable flow problem. While the above approaches concentrate on the Test Bus architecture, Goel and Marinissen [8] proposed an efficient heuristic *TR-Architect* that supports both Test Bus and TestRail architectures. They also presented the lower bounds on SOC test application time in this work. In [10], the same authors extended their algorithm to minimize both test time and TAM wire length. Later in [9], they discussed the influence of test control on test architecture optimization. Given the more practical test pin constraint N_p instead of the total TAM width constraint W_{tam}, the TR-Architect is extended again to a control pin-constrained version.

There have also been plenty of approaches proposed for the flexible-width test architectures. Huang *et al.* [12] first mapped the test architecture optimization to the well-known two-dimensional (2-D) bin packing problem and proposed a heuristic method based on the Best Fit algorithm. Iyengar *et al.* [18] presented another heuristic for this rectangle packing problem, assuming cores are supplied with fixed-length scan chains. By exploiting the feature of Pareto-optimal TAM widths for these cores and through a series of optimization steps, their method was shown to be more effective. Next in [17], they extended their algorithm to incorporate precedence and power constraints, while allowing a group of tests to be preemptable. Later in [20], the same authors considered to optimize the tester buffer reloads and multisite testing, again by extending the rectangle packing algorithm. Zou *et al.* [40] used *sequence pairs* to represent the placement of the rectangles, borrowed from the place-and-route literature. They then employed simulated annealing to find an optimal test schedule by altering an initial sequence pair and rectangle transformation. Su and Wu [36] proposed a graph-based approach to solve the power-constrained test scheduling problem using a tabu search based heuristic for rapid exploration.

It is important to note that all of the above approaches assume static TAM partitions, i.e., the TAM width assigned to each core is fixed during test. Koranne [22] first described a design of reconfigurable core wrapper that allows a dynamic change in the TAM width while executing the core test. This reconfigurability may lead to a more efficient test schedule. Larsson and Peng [25] showed the test scheduling problem to be equivalent to independent job scheduling on identical machines, when reconfigurable wrappers and preemptive scheduling are used. Although reconfigurable wrappers lead to efficient test schedules, more gate and routing overheads are incurred. Moreover, reconfigurability also increases the control complexity of the wrapper. Therefore, in this paper, we only consider static TAM partitions.

B. Motivation and Summary of Contributions

The common feature of the relevant research approaches, summarized in the previous section, is the assumption that the cores' internal scan chain frequencies (f_{tam}) are the same as the ATE's external operational frequency (f_{ext}). In addition, none of the above approaches accounts for the SOC hierarchy when designing TAMs. A limited number of research approaches have been presented to address the above issues separately in [33] and [35] (for multifrequency TAM design) and in [2], [3], [6], [14], [26], and [34] (for hierarchical SOC testing). Prior to outlining our contributions, we will summarize the relevance and limitations of these methods.

Prior Work on Multifrequency TAM Design: The mismatch between the frequencies of the external ATE and internal design for test (DFT) logic leads to underutilization of the available resources, which, in turn, will affect the cost of test. To address this problem, Khoche [21] proposed the *bandwidth matching* technique. *Bandwidth* is defined as the product of the width and the frequency of a scan architecture. Using serialization/deserialization frequency converters, a high bandwidth source/sink (e.g., ATE) can be connected to multiple low bandwidth sinks/sources (e.g., TAMs) as long as the bandwidth matches. As a follow up, Sehgal *et al.* [35] proposed to match ATE channels with high data rates to low-speed core scan chains using virtual TAMs. These virtual TAMs, however, are working at the same frequency. Later in [33], the ATE channels with high data rates are used to directly drive SOC TAM wires, and the heuristic approach based on rectangle packing from [18] was extended to optimize the dual-speed TAM architecture. In order to fully exploit the capability of state-of-the-art ATEs to drive different channels at different data rates, [33] and [35] were proposed. Most of the ATEs currently in use, however, do not have this feature and therefore cannot employ the proposed techniques.

In fact, multifrequency TAM design has the benefits to reduce SOC test cost even for low-end ATEs. This is because, in core-based SOC framework, the scan frequency for an embedded core is variable only within a dedicated range, which is often different for various cores. When single-frequency TAM design is utilized, f_{tam} is a compromise among the frequency ranges of all embedded cores and the external ATE. Since f_{tam} has a direct impact on test application time, TAM wire length, and test power of the SOC, if integrated into test architecture optimization, the system integrator has a larger solution space to explore the tradeoff between these factors, and hence, a better solution can be achieved. Moreover, there may be cases when frequency ranges of multiple cores have no intersection at all, thus constraining system integrators to design multifrequency TAMs to supply data to these cores at different rates. Therefore, prior to tackling the hierarchical SOC test problem, we first investigate a more general multifrequency TAM design approach in a flattened SOC framework.

Prior Work on Hierarchical SOC Testing: Only limited work has been done for testing SOCs that contain megacores. Benso *et al.* [3] proposed a hierarchical-distributed-data built-in self-test (BIST) $(\mathrm{HD^2BIST})$ TAM architecture, which allows test access through design hierarchies. Another TAM architecture presented by Benabdenbi *et al.*, called CAS-BUS (including a core access switch (CAS) and a test bus) [2], also considered the existence of hierarchical megacores. Li *et al.* [26] presented a hierarchical test scheme for SOCs with heterogeneous core tests, in which the proposed hierarchical test manager (HTM) is able to handle megacore testing. All the above articles just briefly discussed how to provide test access for megacores. Recently, two P1500-compliant wrapper architectures for hierarchical cores were proposed, in which megacore level TAM optimization was addressed. Goel [6] introduced a new wrapper cell design that allows a core to operate in INTEST and EXTEST mode concurrently, and hence, allows parallel testing of cores at different levels of hierarchy. His method resulted in reduced testing time for hierarchical SOCs at the cost of a larger DFT area overhead. Sehgal *et al.* [34] presented a general architecture for hierarchical core wrappers using conventional P1500 wrapper cells [31] and described various modes of operation of the wrapper. The proposed wrapper design is reconfigurable in order to operate efficiently in all the test modes. In [6] and [34], megacore level TAMs are assumed to be "soft," i.e., they are not fixed and system integrators are in charge of their design and optimization. In this article, however, we consider the case that megacore level TAMs are "hard," i.e., their implementation is fixed

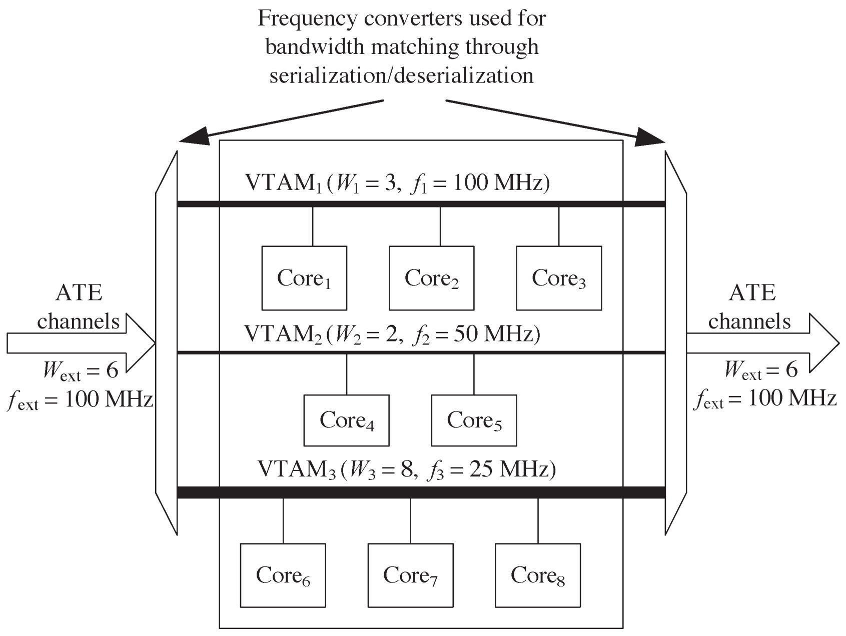

Frequency converters used for bandwidth matching through serialization/deserialization
VTAM$_1$ ($W_1 = 3$, $f_1 = 100$ MHz)
Core$_1$
Core$_2$
Core$_3$
VTAM$_2$ ($W_2 = 2$, $f_2 = 50$ MHz)
Core$_4$
Core$_5$
VTAM$_3$ ($W_3 = 8$, $f_3 = 25$ MHz)
Core$_6$
Core$_7$
Core$_8$
ATE channels
$W_{ext} = 6$
$f_{ext} = 100$ MHz
ATE channels
$W_{ext} = 6$
$f_{ext} = 100$ MHz

Fig. 2. Proposed multifrequency test architecture for flattened SOCs.

"as is." Not only is the TAM architecture inside predesigned, but the test schedule of all its child cores is also predetermined. The test of the megacore itself is also implemented by the core provider (unlike in [6] and [34] where it is implemented by system integrators), which could be a megacore level EXTEST (similar to SOC EXTEST to test interconnecting logic) or a "hard" implementation of the test strategy proposed in [6] and [34].

In another approach to hierarchical SOC testing [14], Iyengar *et al.* presented how to extend known wrapper/TAM optimization methods for flattened SOCs [15], [16] to multilevel TAM optimization in two different design transfer models. For the *interactive* design transfer model (i.e., the core provider can change the TAM based on the request of system integrators), a set of megacore instances with different TAM widths need to be delivered, which obviously affects the reuse methodology and inhibits the development of hard megacores. To address this issue, *noninteractive* design transfer model can be employed (i.e., the cores are taken off-the-shelf and integrated into designs "as is"). However, in this case, the system integrator has to design a wider system-level TAM to fork out to the predesigned core-level TAMs. In addition, the constrained TAM width of the megacore leads to very inflexible test schedule, and hence, increased test application time.

Based on the above, in this paper, we explore the suitability of exploiting multifrequency test data transportation to lower the test application time in hierarchical SOCs with hard megacores. The two main contributions of this paper, detailed in Sections III and IV, are as follows.

1) We first present a new multifrequency TAM design algorithm for flattened SOCs; unlike [33] and [35], not only do we consider the case when the ATE is faster than the scan chains, but we also examine the case when low-speed ATEs are used for high-speed scan chains, which may also reduce the routing overhead as long as power ratings are not exceeded; this is achieved by matching the bandwidth at the TAM level and refining an effective exploration engine [8], which, ultimately, reduces test application time.
2) We then extend the multifrequency concepts to a multilevel TAM design algorithm for hierarchical SOCs; to fully reuse the hard megacores in a noninteractive design transfer model, we examine the usage of two types of frequency converters that can match a higher number of core-level TAM lines to a lower number of system-level TAM lines; thereafter, by proposing a new design flow, in contrast to [14], we provide the system integrator the option to trade the DFT area against savings in test application time.

III. Multifrequency TAM Design for Flattened SOCs

Prior to addressing the test of hierarchical SOCs, we first explain how multifrequency test data transportation can reduce the testing time for flattened SOCs (i.e., all the embedded cores are on the same level of hierarchy). We start by describing the architecture, then we formulate the problem to be solved and propose an extension of TR-Architect [8] to support multifrequency TAMs. Finally, we illustrate the benefits of the proposed approach on a benchmark SOC [30].

A. Multifrequency Test Architecture

Fig. 2 shows the proposed test access architecture for flattened SOCs. To match the bandwidth of six external tester channels running at 100 MHz to three internal TAM lines running at 100 MHz, two internal TAM lines running at 50 MHz, and eight internal TAM lines running at 25 MHz, we place serialization/deserialization frequency converters at the input/output (I/O) of the TAM groups. To keep the area

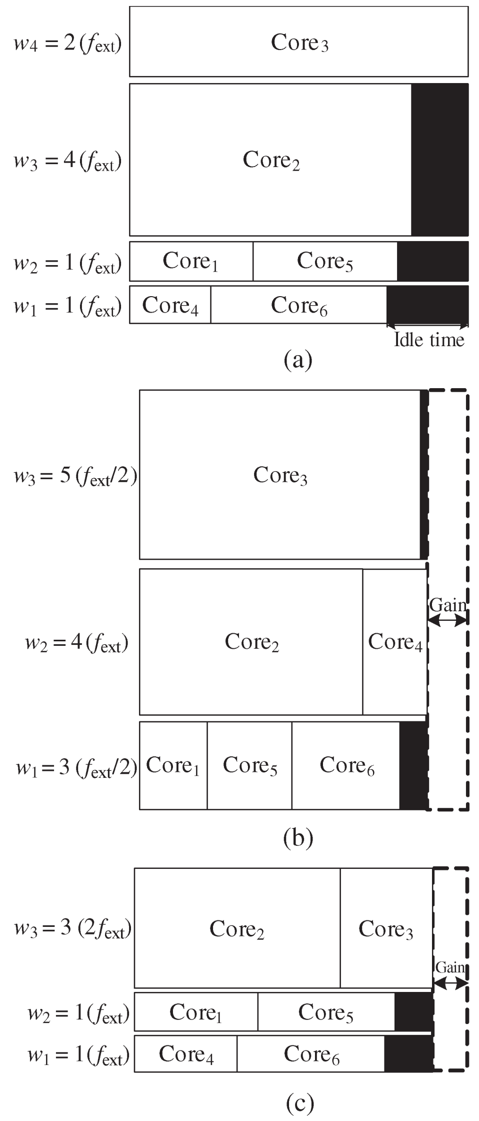


Fig. 3. Motivational example showing the benefits of multifrequency TAM design. (a) $f_{\rm tam} = f_{\rm ext}$; (b) $f_{\rm tam} \leq f_{\rm ext}$; (c) $f_{\rm tam} \leq 2f_{\rm ext}$.

overhead low and to facilitate our proposed algorithm (detailed in Section III-B), we consider that the relationship between frequencies is a power of 2. This will lead to a straightforward serial/parallel shift register implementation for the frequency converters. Although the ratio between any two frequencies is a power of 2, this does not necessarily imply that the ratio between the number of tester channels and any TAM group is a power 2, since we can have separate TAM groups working at the same frequency. This is illustrated in the following example that shows the advantage of using multiple frequencies for test data transportation.

Example 1: Consider a hypothetical SOC with six cores. The TAMs for these six cores are on the same level and the test schedules are shown in Fig. 3. The original test schedule (with all TAMs operating at $f_{\rm tam} = f_{\rm ext}$) is shown in Fig. 3(a), where four TAMs are employed (w_i is the width of TAMi) and TAM4 is the bottleneck TAM, i.e., the overall test application time is dominated by it. It can be seen that the idle time for the nonbottleneck TAMs is large. If, however, multifrequency virtual TAMs are designed using the bandwidth matching technique with $f_{\rm tam} \leq f_{\rm ext}$ (low-frequency $f_{\rm tam}$ synchronized to a divisor of $f_{\rm ext}$), as shown in Fig. 3(b), TAM1, TAM2, and TAM3 operate at $f_{\rm ext}/2$, $f_{\rm ext}$, and $f_{\rm ext}/2$, respectively. The original bottleneck TAM takes less time with $w_3 = 5$ operating at $f_{\rm ext}/2$, and the overall test application time is decreased with the new bottleneck TAM2. Unlike in [35], the serial/parallel converters used for bandwidth matching are placed at the TAM level, which may introduce additional routing overhead, however, it will lead to more flexibility in the test schedule (i.e., the number of low-speed TAM lines does not need to be a multiple by a power of 2 of the number of high-speed tester lines). For example, in Fig. 3(b), the width of the TAM bus used by Core$_3$ operating at $f_{\rm ext}/2$ is $w_3 = 5$. Note that since the virtual TAMs are generated internally, the number of SOC pins for test purpose remains the same. If the TAMs can operate at a frequency higher than that of the ATE's ($f_{\rm tam} \leq 2f_{\rm ext}$ in this example) and the power ratings are not exceeded, then both the testing time and the routing overhead of the SOC can be decreased, as shown in Fig. 3(c). This also facilitates high-speed scan testing (scan frequency at several hundreds of megahertz as shown in [11] and [38]) using low-end ATEs, and, in terms of design methodology, the main difference to low-speed testing using high-speed ATEs lies in using parallel/serial conversion on the input and serial/parallel conversion on the output. When using low-speed ATEs for high-speed scan chains, it is assumed that a high-speed clock, generated on-chip, is used to serialize the low-speed data arriving from the ATE.

B. Multifrequency TAM Design Algorithm

The multifrequency TAM design problem, addressed in this section, is formulated as follows.

Problem $P_{\rm mf-TAM-opt}$: Given the maximum TAM width $W_{\rm max}$ for the SOC, the operating frequency of the ATE $f_{\rm ext}$, the test set parameters for each core, including the number of input terminals, the number of output terminals, the number of test patterns, the number of scan chains, and, for each scan chain, its length (for cores with fixed-length internal scan chains) or the total number of scan flip-flops (for cores with flexible-length internal scan chains), determine the width of each virtual TAM, the shift frequency $f_{\rm tam}$ of each virtual TAM, the wrapper design for each core, and a test schedule for the entire SOC such that 1) the bandwidth of the internal TAM used at any time does not exceed the bandwidth of the ATE and 2) the overall SOC test application time is minimized.

Because the single-frequency TAM design problem, which is known to be NP-hard [4], is a special case of the above one, in the following, we proposed a heuristic algorithm to solve it. In our implementation, we have adapted an existing effective algorithm (TR-Architect [8]) to multifrequency TAM design, which is able to support both the TestRail and Test Bus architectures. The wrapper design algorithm that we used in this paper, for cores with fixed-length scan chains, is the Combine procedure from [28]. For flexible-length scan chains, we have assumed that they are balanced in wrapper optimization.

Data Structure: The data structure TAM contains four elements. $T_{\rm tam}$ is measured in ATE clock cycles. We define the frequency ratio of the TAM as $\mathrm{freqRatio} = f_{\mathrm{ext}}/f_{\mathrm{tam}}$. Suppose the test application time of the TAM in its operating frequency is T_{of} clock cycles, then $T_{\mathrm{tam}} = T_{\mathrm{of}} \times \mathrm{freqRatio}$. This data structure is updated whenever TAM merging, freed wires assignment, or core test reshuffle during test scheduling happens.

```
Data Structure TAM
1. width$(i)$/* width of TAM $i$*/
2. coreList$(i)$/* List of cores tested on TAM $i$*/
3. freqRatio$(i)$/* The freq. ratio between tester and TAM $i$*/
4. $T_{\mathrm{tam}}(i)$/* Test application time of TAM $i$*/
```

Proposed Top-Level Algorithm: In mfTAMDesign [shown in Algorithm 1] we first compute the initial virtual TAM width using the bandwidth matching technique. Then we iteratively initialize the virtual TAM width W_{vt} as 2's exponent of the initVTWidth (line 4). N_{mul} is a constant to stop the search in solution space (in our experiments, $N_{\mathrm{mul}} = 8$ provides a good tradeoff between the computation time, which is at most within a few seconds even for large SOCs, and the quality of the final solution in terms of test application time). The maximum frequency ratio (maxFreqRatio) constrains the lowest possible frequency $f_{\min} = f_{\mathrm{ext}}/\mathrm{maxFreqRatio}$ for current virtual TAM width (line 5). In line 6, we call an adapted TR-Architect algorithm mf-TR-Architect to get the test application time $T_{\mathrm{soc}}(i)$ with current virtual TAM width W_{vt}. When compared to [8], the main differences are in merging multifrequency TAMs and distributing freed TAM resources, which are detailed in Algorithms 2 and 3. Since a large number of virtual TAMs may lead to routing overhead, we use a variable layoutconstraint to restrict the number of virtual TAMs $W_{\mathrm{soc}} \leq \mathrm{layoutconstraint} \times W_{\max}$, and we choose the TAM design $\mathrm{TAM}_{\mathrm{design}}$ with the minimum T_{soc} without exceeding the layout constraint W_{soc}. In this paper, we assume that all the channels of the ATE run at the same frequency f_{ext}, however, if the tester has channels with different speeds, by applying the bandwidth matching translations, the number of ATE channels visible by the SOC can be recalculated and provided as input to our algorithm.

```
Algorithm 1—mfTAMDesign
Input: $C$, layoutConstraint, $W_{\max}$
Ouput: $\mathrm{TAM}_{\mathrm{design}}, T_{\mathrm{soc}}$
 1. COMPUTE initVTWidth;
 2. ASSIGN multiple $= 1$;
 3. for i from 1 to $N_{\mathrm{mul}}$ {
 4.    $W_{\mathrm{vt}} = \mathrm{multiple} \times \mathrm{initVTWidth}$;
 5.    $\max\mathrm{FreqRatio} = \mathrm{multiple}$;
 6.    $T_{\mathrm{soc}}(i) = \mathrm{mf} - \mathrm{TR} - \mathrm{Architect}(W_{\mathrm{vt}},$
         $\max\mathrm{FreqRatio}, C)$;
 7.    $T_{\mathrm{soc}} = \min($all $T_{\mathrm{soc}}(i)$ with
         $W_{\mathrm{soc}(i)} <=$ layoutConstraint $\times W_{\max})$;
 8.    Record $\mathrm{TAM}_{\mathrm{design}}$ with testing time $T_{\mathrm{soc}}$;
 9.    multiple $=$ multiple $\times 2$;
  . }
10. RETURN $(T_{\mathrm{soc}},\ \mathrm{TAM}_{\mathrm{design}})$
```

TR-Architect Algorithm [8]: TR-Architect has four main steps. The basic idea is to divide the total TAM width over multiple cores based on their test data volume. The algorithm first create an initial test architecture in the procedure CreateStartSolution by assigning the value 1 to each core's TAM width. Since the overall test application time of the SOC T_{soc} equals the bottleneck TAM with the longest test application time, in the second (procedure Optimize-BottomUp) and the third (procedure Optimize-TopDown) steps, the algorithm iteratively optimizes T_{soc} through merging TAMs and distributing freed TAM resources. Either two nonbottleneck TAMs are merged with less TAM width to release freed TAM resources to the bottleneck TAM, or the bottleneck TAMs are merged with another TAM to decrease T_{soc}. In the last step (procedure Reshuffle), the algorithm tries to further minimize T_{soc} by moving one of the cores from the bottleneck TAM to another TAM. It should be noted that we only show the differences with respect to the original algorithm in the following two paragraphs.

Merging Multifrequency TAMs: The procedure to merge multifrequency TAMs is shown in Algorithm 2. It enumeratively merges TAMs with different frequency/width configuration and selects the one with minimum T_{tam}. The algorithm starts by merging the cores on TAM1 and TAM2 (line 1) and then the frequency ratio $\mathrm{freqRatio}_{\mathrm{tam}}$ and the width of the TAM W_{tam} are initialized to maxFreqRatio and vtWidth, respectively (lines 2 and 3). In the loop (lines 4–9), we enumeratively change the frequency/width configuration and compute $T_{\mathrm{tam}}(\mathrm{freqRatio}_{\mathrm{tam}})$. Finally, the TAM configuration mergedTAM with minimum T_{tam} is returned.

```
Algorithm 2—mergeMFTAMs
Input: TAM1, TAM2, vtWidth
Output: mergedTAM, $T_{\mathrm{tam}}$
 1. $\mathrm{coreList}_{\mathrm{tam}} = \mathrm{coreList}_{\mathrm{tam1}} \bigcup \mathrm{coreList}_{\mathrm{tam2}}$;
 2. ASSIGN $\mathrm{freqRatio}_{\mathrm{tam}} = \mathrm{maxFreqRatio}$;
 3. ASSIGN $W_{\mathrm{tam}} = \mathrm{vtWidth}$;
 4. FOR all frequency ratios {
 5.    COMPUTE $T_{\mathrm{tam}}(\mathrm{freqRatio}_{\mathrm{tam}})$;
 6.    $T_{\mathrm{tam}} = \min\{$all $T_{\mathrm{tam}}(\mathrm{freqRatio}_{\mathrm{tam}})\}$;
 7.    Record mergedTAM with minimum testing time $T_{\mathrm{tam}}$;
 8.    $\mathrm{freqRatio}/ = 2$;
 9.    $W_{\mathrm{tam}}/ = 2$;
  . }
10. RETURN (mergedTAM, $T_{\mathrm{tam}}$)
```

Distributing Free Virtual TAM Resources: Since the proposed solution does not restrict the number of lines in a TAM group working at a lower frequency to be a power 2 [see, for example, w_1 and w_3 in Fig. 3(b)], there is added flexibility for TAM merging, which ultimately turns out in more free virtual TAM resources that can be used for test application time reduction. The procedure distributeVT, shown in Algorithm 3, is used to iteratively distribute virtual TAM resources to reduce the idle time. The algorithm iteratively finds the bottleneck TAM (line 2) and then computes the wire width (wireWidth) of this virtual TAM in terms of virtual TAM lines operating at $f_{\min}$ (line 3). Inside the FOR loop (lines 6–29), the algorithm tries to

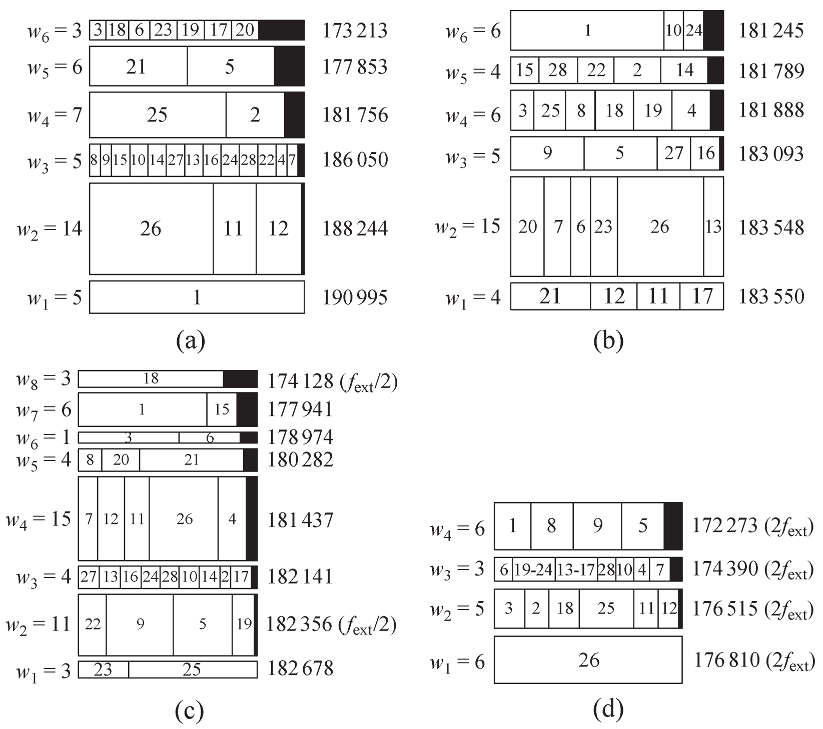

w6 = 3 | 3 18 6 23 19 17 20 | 173 213
w5 = 6 | 21 5 | 177 853
w4 = 7 | 25 2 | 181 756
w3 = 5 | 8 9 15 10 14 27 13 16 24 28 22 4 7 | 186 050
w2 = 14 | 26 11 12 | 188 244
w1 = 5 | 1 | 190 995
(a)
w6 = 6 | 1 10 24 | 181 245
w5 = 4 | 15 28 22 2 14 | 181 789
w4 = 6 | 3 25 8 18 19 4 | 181 888
w3 = 5 | 9 5 27 16 | 183 093
w2 = 15 | 20 7 6 23 26 13 | 183 548
w1 = 4 | 21 12 11 17 | 183 550
(b)
w8 = 3 | 18 | 174 128 (fext/2)
w7 = 6 | 1 15 | 177 941
w6 = 1 | 3 6 | 178 974
w5 = 4 | 8 20 21 | 180 282
w4 = 15 | 7 12 11 26 4 | 181 437
w3 = 4 | 27 13 16 24 28 10 14 2 17 | 182 141
w2 = 11 | 22 9 5 19 | 182 356 (fext/2)
w1 = 3 | 23 25 | 182 678
(c)
w4 = 6 | 1 8 9 5 | 172 273 (2fext)
w3 = 3 | 6 19-24 13-17 28 10 4 7 | 174 390 (2fext)
w2 = 5 | 3 2 18 25 11 12 | 176 515 (2fext)
w1 = 6 | 26 | 176 810 (2fext)
(d)

Fig. 4. Comparison of TAM design for flattened SOC p22810 with $W_{\max} = 48$. (a) TR-Architect; (b) mfTAMDesign ($f_{\text{tam}} = f_{\text{ext}}$); (c) mfTAMDesign ($f_{\text{tam}} \le f_{\text{ext}}$); (d) mfTAMDesign ($f_{\text{tam}} \le 2f_{\text{ext}}$).

assign the least number of virtual TAM wires that can reduce T_{soc}. Whenever an additional virtual TAM wire is assigned to the bottleneck TAM, the algorithm tries to find whether the test application time can be decreased with different configuration of frequency and TAM width (lines 14–23). If T_{soc} is reduced (lines 24–26), the algorithm will loop back to find the new bottleneck TAM and start again. Even if T_{soc} is not decreased by assigning wireWidth of virtual TAM lines, the algorithm will still distribute this number of virtual TAM lines to the bottleneck TAM (lines 27–29) as this may provide a good starting point for the next iteration within TR-Architect [8]. The algorithm terminates when $\text{freedVT} = 0$ or the test application time of the bottleneck TAM cannot be decreased and at the same time $\text{freedVT} < \text{wireWidth}$ (line 30). The freed virtual TAM wires $\text{freedVT}'$ can be used within the future iterations of TR-Architect to decrease T_{soc}. The benefits of using multifrequency TAMs are illustrated next.

Algorithm 3—distributeVT
Input: R, freedVT
Output: R', $\text{freedVT}'$

```
1.  WHILE ($\text{freedVT} > 0$) {
2.    find $r_{\max}$ with maximum $T_r$;
3.    $\text{wireWidth} = \text{maxFreqRatio} \div \text{freqRatio}_{r_{\max}}$;
4.    $r_{\text{temp}} = r_{\max}$;
5.    assign $\text{isImproved} = \text{false}$;
6.    FOR($\text{width} = 1$; $\text{width} <= \text{wireWidth}$; $\text{width} *= 2$) {
7.      IF($\text{freedVT} < \text{width}$) BREAK;
8.      $\text{freqRatio}_{r_{\text{temp}}} = \text{maxFreqRatio}$;
9.      $\text{width}_{r_{\text{temp}}} += \text{width}$;
10.     compute $\text{minTime} = T_{r_{\text{temp}}}$;
11.     IF($T_{r_{\text{temp}}} < T_{r_{\max}}$) {
12.       $r_{\max} = r_{\text{temp}}$;
13.       $\text{isImproved} = \text{true}$;}
14.     WHILE (($\text{freqRatio}_{r_{\text{temp}}} \ge 2$)&&
              ($\text{width}_{r_{\text{temp}}} \% 2 == 0$)) {
15.       $\text{freqRatio}_{r_{\text{temp}}} /= 2$;
16.       $\text{width}_{r_{\max}} /= 2$;
17.       compute $T_{r_{\text{temp}}}$;
18.       IF (($T_{r_{\text{temp}}} < T_{r_{\max}}$)&&($T_{r_{\text{temp}}} < \text{minTime}$)) {
19.         $\text{minTime} = T_{r_{\text{temp}}}$;
20.         $r_{\max} = r_{\text{temp}}$;
21.         $\text{isImproved} = \text{true}$;
22.       }ELSE {
23.         continue; }
.       }
24.     IF ($\text{isImproved} == \text{true}$) {
25.         $\text{freedVT} -= \text{width}$;
26.         BREAK;
27.     }ELSE IF ($\text{width} == \text{wireWidth}$) {
28.       $\text{width}_{r_{\max}} += 1$;
29.       $\text{freedVT} -= \text{width}$; }
.     }
30.   IF ($\text{freedVT} < \text{wireWidth}$) &&
        (!isImproved) BREAK;
.  }
31. $\text{freedVT}' = \text{freedVT}$;
32. RETURN ($R'$, $\text{freedVT}'$)
```

C. Case Study for Benchmark SOC p22810

Fig. 4 shows the schedules for the benchmark SOC p22810 [30] with TR-Architect and mfTAMDesign algorithm, assuming fixed-length scan chains, the Test Bus architecture, and a total TAM width of 40. With TR-Architect [Fig. 4(a)], six TAMs are obtained and the maximum test application time is 190 995 clock cycles. When using mfTAMDesign with $f_{\text{tam}} = f_{\text{ext}}$, a more balanced TAM design (less idle time) is achieved (183 550 clock cycles) without introducing any hardware overhead [Fig. 4(b)]. This proves that embedding the proposed multifrequency TAM exploration in TAM merging engine of TR-Architect [8], savings in test application time can be achieved in the single-frequency case. When additional routing overhead is affordable, the test application time can be further decreased to 182 678 clock cycles, in which eight TAMs are designed with seven additional virtual TAM lines [Fig. 4(c)]. When the power ratings allow the scan chains to run at $2f_{\text{ext}}$, 176 810 clock cycles are obtained with only 20 internal virtual TAM lines [Fig. 4(d)]. Frequency converters, implemented as shift registers, are needed for each of the 14 low-speed virtual TAM lines working at $f_{\text{ext}}/2$ in Fig. 4(c) and each of the 20 high-speed virtual TAM lines working at $2f_{\text{ext}}$ in Fig. 4(d).

IV. Multifrequency TAM Design for Hierarchical SOCs

In this section, we show how multifrequency test data transportation can facilitate the reuse of hard megacores in a noninteractive design transfer model, without a negative impact on test application time. We first examine two types of frequency converters that can match a higher number of core-level TAM lines to a lower number of system-level TAM lines. Then we introduce a new design flow for the system integrator, and we illustrate the benefits of the proposed approach on a hierarchical benchmark SOC [30].

A. Matching the Bandwidth for Hard Megacores

Suppose the core-level TAM width within the megacore is W_l, which operates at frequency f_l, and the external system-level TAM width W_h is assigned to the megacore, which operates at f_h (not necessarily the same as ATE frequency f_{ext}). When $W_h = W_l$, the system-level TAM connects to the core-level TAM directly with the same frequency. If $W_h < W_l$, by introducing W_h/W_l (W_l/W_h) frequency converters on the input (output) of the megacore, the core-level TAM operates at a lower frequency $f_l = (f_h W_h / W_l)$, as shown in Fig. 5. Based on the relationship between W_h and W_l, we define the two types of converters used in this paper as follows:

1) *Type I converters* are pairs of shift registers and the corresponding control logic for bandwidth matching. They are used when W_h is a divisor of W_l.
2) *Type II converters* are pairs of buffers with depth $(W_l W_h/\text{gdiv})$ and the corresponding control logic for bandwidth matching, where gdiv is the greatest common divisor of W_l and W_h. They are used when W_h is not a divisor of W_l.

In Example 2, we show the implementation of a type I 1/4 (4/1) frequency converter and a type II 3/4 (4/3) frequency converter and analyze the impact to use these frequency converters for SOC testing. It should be noted that, for a general

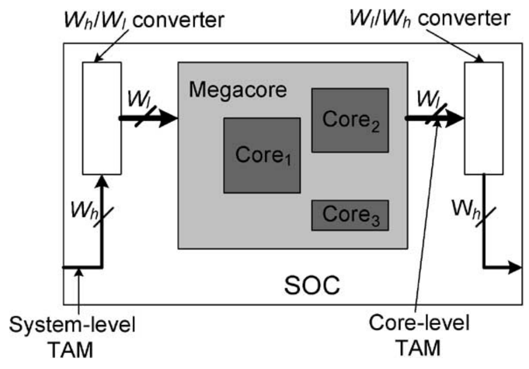


Fig. 5. Proposed hardware architecture for matching the bandwidth for hard megacores.

W_h/W_l (W_l/W_h) frequency converter, the hardware implementation is based on the same concepts outlined in the following example.

Example 2: Consider a hard megacore with core-level TAM width $W_l = 4$. The test application time is T core clock cycles.

1) If the system-level TAM width assigned to the megacore is $W_h = 1$, then the type I 1/4 (4/1) frequency converter can be implemented as shown in Fig. 6(a). For the shift register implementation, only eight flip-flops and a clock division unit are needed to map the one system-level TAM line to the four core-level TAM lines, in which Clock Div. [see Fig. 6(a)] generates the megacore wrapper test clock signal TCK and also the mux control signal Mux_Sel for the 4/1 frequency converter. The test application time for the megacore will be $4T$ system clock cycles (since the core-level TAM operates at $f_h/4$).
2) If, however, three system-level TAM lines are assigned to this megacore $(W_h = 3)$, then the type II 3/4 (4/3) frequency converters can be implemented as shown in Fig. 6(b). Two 12×1 register array serves to transfer data between the system-level TAM and core-level TAM. The Ctrl-FSM is a finite state machine that counts the number of writes/reads to/from the buffer and controls the increment, decrement, or hold of the buffer address. It also generates the TCK signal with frequency $f_l = (3/4)f_h$. The test application time for the megacore will be $(4/3)T$ system clock cycles, however, this type II converter takes more area than type I converter.

For both type I and type II frequency converters, the size of the buffer to map test data from the system-level TAM to the core-level TAM is $S_{\text{buffer}} = \text{LCM}(W_h, W_l) \times 2$, where W_h is the system-level TAM width assigned to the megacore, W_l is internal TAM width of the megacore and LCM stands for the least common multiplier. For the control logic part, if the small combinational logic inside is ignored, type I converter requires W_h/W_l flip-flops, while type II converter requires $\lceil \log S_{\text{buffer}} \rceil$ flip-flops, which is generally small when compared to the buffer size. As a result, we use S_{buffer} as the added DFT area constraint of the system-level exploration algorithm proposed later in this section.

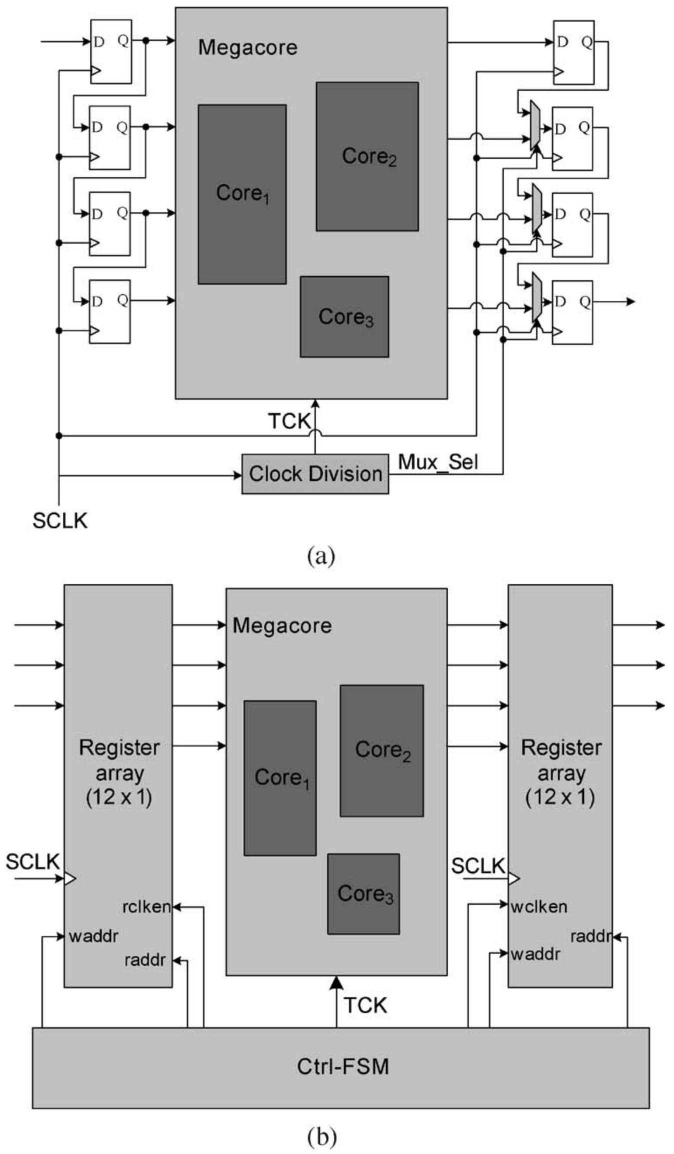


Fig. 6. Type I and II frequency converters used for bandwidth matching. (a) 1/4 and 4/1 type I converters; (b) 3/4 and 4/3 type II converters.

Prior to introducing the design flow for the system integrator, we show, using an example, the benefits to use type I and II converters for hierarchical SOC testing.

Example 3: Consider a hypothetical hierarchical SOC having two system-level cores Core_8 and Core_9 and two system-level megacores Core_1 and Core_5 (labeled as 1–4 and 5–7) as shown in Fig. 7. Suppose the megacore provider has implemented "hard" (i.e., nonalterable) TAMs within the megacores and the width of the internal TAM (level-2 TAM in this example) for both Core_1 and Core_5 is 8. If the system integrator is constrained to use a system-level TAM width of 12, the test schedule using [14] is shown in Fig. 7(a), where the two megacores need to be tested sequentially and compose the bottleneck TAM of the SOC (i.e., the overall test application time is dominated by TAM2). However, if we introduce type I converters next to the I/Os of the megacore wrappers, in order to divide by 2 the system-level TAM width assigned to megacore Core_5, the overall test application time will be reduced because

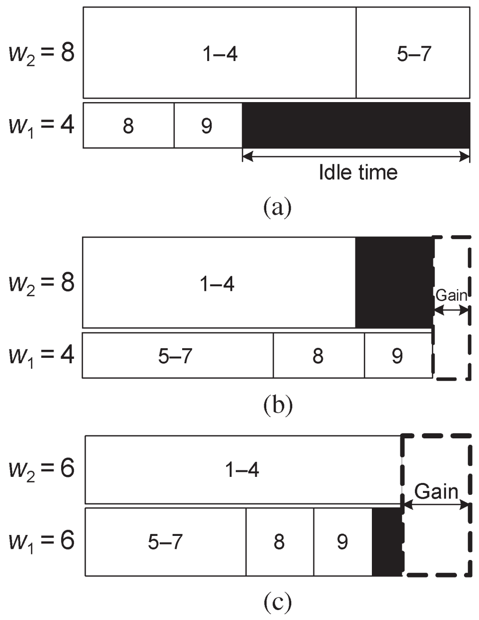


Fig. 7. Benefits of multifrequency TAM design for hierarchical SOCs. (a) No bandwidth matching. (b) Type I converters. (c) Type II converters.

Core_5 is placed in the same TAM group as Core_8 and Core_9. This case is shown in Fig. 7(b), and it can be seen that TAM1 has become the bottleneck TAM. If type II converters are used for both Core_1 and Core_5, the test application time can be further decreased, as shown in Fig. 7(c), because six system-level TAM lines are assigned to each megacore. However, two pairs of 6/8 and 8/6 type II converters are required, which leads to significantly more area than in the case of Fig. 7(b), where a single pair of 4/8 and 8/4 type I converters is required.

While it is clear that converters of both types can be employed to reduce test application time when hard megacores are used, the question is how much DFT area needs to be introduced? Since type I converters incur less overhead than type II converters, in the following section, we introduce a new design flow for the system integrator, which facilitates an easy tradeoff between the reduction in test application time and the DFT area necessary to achieve it.

B. Design Flow for System Integrator

DFT area to implement frequency converters is considered as a constraint to the multilevel TAM design problem, which is defined next.

Problem $P_{\mathrm{ml-TAM-opt}}$*:* Given the maximum system-level TAM width $W_{\max}$ for the hierarchical SOC, the DFT area overhead constraint C_{area} to implement frequency converters, the test set parameters for each top-level core, including the number of input terminals, the number of output terminals, the number of test patterns, the number of scan chains, and, for each scan chain, its length (for cores with fixed-length internal scan chains) or the total number of scan flip-flops (for cores with flexible-length internal scan chains), the test parameters for each "hard" megacore, including the prespecified core-level

TAM width W_l and its test application time T for its child cores and the megacore itself, determine the width and shift frequency of each system-level TAM, the wrapper design for each core, the test resources to map the SOC system-level TAM to each core-level TAM within the megacore, and a test schedule for the entire SOC, such that 1) the bandwidth of the internal TAM used at any time does not exceed the bandwidth of the ATE; 2) the total buffer size of all the converters used for megacores does not exceed C_{area}; and 3) the overall SOC test application time is minimized.

The basic intuition behind the proposed solution to the design of multilevel TAMs is to trade the test application time and the type and size of the frequency converters used to match the bandwidth between the SOC TAM lines and the hardwired megacores' TAM lines. Once the previous step is completed, megacores can be treated as regular cores and a multifrequency variation of TR-Architect [8], described in Section III, is used to solve the flattened SOC TAM problem. In the following, we explain the cost model necessary for constrained design space exploration and the proposed design flow for the system integrator.

Cost Model for Megacore: Cost Model for Megacore: By building a cost model for each megacore using both test application time and hardware cost, the test scheduling algorithm tries to optimize them simultaneously. For each megacore i, the overall test cost is formulated as $C_i = T_i + S_{\text{buffer\_}i} \times costWeight$, where T_i is its test application time, $S_{\text{buffer\_}i}$ is the buffer size necessary to implement frequency conversion, and costWeight is a weighting factor that is varied by the system integrator during the solution space exploration.

Design Flow for the System Integrator: The design flow is shown in Algorithm 4. It starts by allowing both type I and type II frequency converters to be used (line 2). Inside the internal loop (lines 5–25), the algorithm tries to find the schedule with the lowest test application time while fulfilling the area constraint. If the constraint cannot be met, then the algorithm will try again only with type I converters (line 24). If the constraint still cannot be met, there will be no converters that can guarantee that area constraint is met and the design flow will become, in principle, the same as in [14] (lines 27–32). Finally, the system-level TAM architecture, which can meet the area constraint C_{area} with the lowest test application time, is implemented (line 33). It is important to note that even when DFT area constraints are tight, the test scheduling flexibility leads to savings in test application time, as shown in our experiments.

Algorithm 4—Design Flow for the System Integrator

```
 1. get the core test parameters for nonhierarchical cores;
 2. set typeIIAllowed = true; typeIAllowed = true;
 3. set isConstraintMet = false;
 4. WHILE (!isConstraintMet) {
  .  /*Converters are allowed, different area cost weight $w_n$
       are tried*/
 5.   IF(typeIAllowed OR typeIIAllowed){
 6.     FOR each iteration $n$ in $N$ iterations {
 7.       FOR each top-level megacore $i$ in the SOC {
 8.         set $w_n = n \times p$;
 9.         get the prespecified TAM width and test applica-
            tion time;
  .         /* Let $T_i(w_j)$ and $C_i(w_j)$ be the SOC-level test
  .            application time and overall cost of Core $i$ with
  .            TAM width $w_j$*/
10.         set $T_i(w_j) = T_i$, $C_i(w_j) = T_i$, for $w_j \geq W_i$;
11.         FOR $(w_j < W_i)$ {
12.           IF (typeIIAllowed) {
13.             set $T_i(w_j) = (T_i \times W_i/w_j)$;
14.             set $C_i(w_j) = T_i(w_j) + S_{\text{buffer\_i}} \times w_n$; }
15.           ELSE IF (typeIAllowed) {
16.             set $T_i(w_j) = T_i \times N_f$;
17.             set $C_i(w_j) = T_i(w_j) + S_{\text{buffer\_}i} \times w_n$; }
  .         }
  .       }
18.       partition $W_{\max}$ system-level TAM among cores
          using mfTAMDesign;
  .     }
19.     IF (test schedule with BufferSize $\leq C$ exists) {
20.       record the test schedule with minimum time;
21.       isConstraintMet = true;
22.     } ELSE {
23.     isConstraintMet = false;
24.     IF(typeIIAllowed) typeIIAllowed = false;
25.     ELSE typeIAllowed = false; }
26. }ELSE {
  .   /* No converters is allowed */
27.   FOR each top-level megacore $i$ in the SOC {
28.     get the prespecified TAM width and test application
        time;
29.     set $T_i(w_j) = T_i$, $C_i(w_j) = T_i$, for $w_j \geq W_i$;
30.     FOR $(w_j < W_i)$ {
31.       set $T_i(w_j) = \infty$; $C_i(w_j) = \infty$; }
  .   }
32.   partition $W_{\max}$ system-level TAM among cores using
      mfTAMDesign;
  . }
 .}
33. implement system-level TAM architecture;
```

In the following, we place the focus on the internal loop of the algorithm where the frequency converters are used (lines 5–25). Since the internal core-level TAM width W_i cannot be changed, if $w_j \geq W_i$, w_j of the system-level TAM lines connect to the core-level TAM lines directly, and hence, the test application time equals T_i (line 10). When $w_j < W_i$, the test application time of the megacore is dependent on the type of test resources the system integrator can introduce (lines 13–17). Note that if type I converter is introduced, only w of w_j system-level TAM lines can be mapped to the core-level TAM lines, where w is the greatest factor of W_i less than w_j ($N_f = W_i/w$). After we get the information on TAM width/test application time pair for each core or megacore, in line 18, we partition the system-level TAM using the flattened TAM optimization algorithm mfTAMDesign, which was described in the previous section. The cost weight w_n in the overall cost is varied several times in order to explore a high number of solutions (lines 6–17). w_n is computed in line 8, in which p

is a scaling factor used to avoid the usually large difference between the buffer size and the test application time measured in clock cycles. For the results reported in this paper, we have mostly used $p = 50$ and the number of iterations is $N = 20$, which gives good results with computation times in the range of seconds.

C. Case Study for Benchmark SOC p93791

Fig. 8 shows the test schedule for the hierarchical benchmark SOC p93791 with total system-level TAM width of 48 and when the core-level TAM width for each megacore is fixed at 16. When no frequency converters are implemented, the test application time is 801 271 clock cycles, as shown in Fig. 8(a). Although the idle time does not seem to be large, since megacores 14, 17, 20, 27, and 29 are on the TAM with width 26, only 16 system-level TAMs connect to their core-level TAM (while the other 10 system-level TAM lines are wasted, which is obviously inefficient). When the area constraint is set as 100, type I converter for megacores 20 and 29 is used and the test application time is reduced to 631 656 clock cycles. Frequency conversion is achieved using a shift register implementation, which needs 64 flip-flops for the buffer and some control logic. When type II converters for megacores 1, 14, 20, and 29, and type I converter for megacores 27 are employed, the test application time can be further decreased to 611 859 clock cycles, however, with a larger hardware cost. In total, 608 flip-flops are required for the buffers to implement bandwidth matching for the entire SOC. It should be noted that this SOC has a large number of megacores when compared to other designs, which is the main source of the increased test area caused by type II converters. Since, in principle, a megacore is supposed to be large and the top-level SOC that includes it should be even larger, we consider that the area penalty introduced by the type II converters can be acceptable, especially when ATE buffer depth is low and reductions in test application time are essential to avoid ATE buffer reloading.

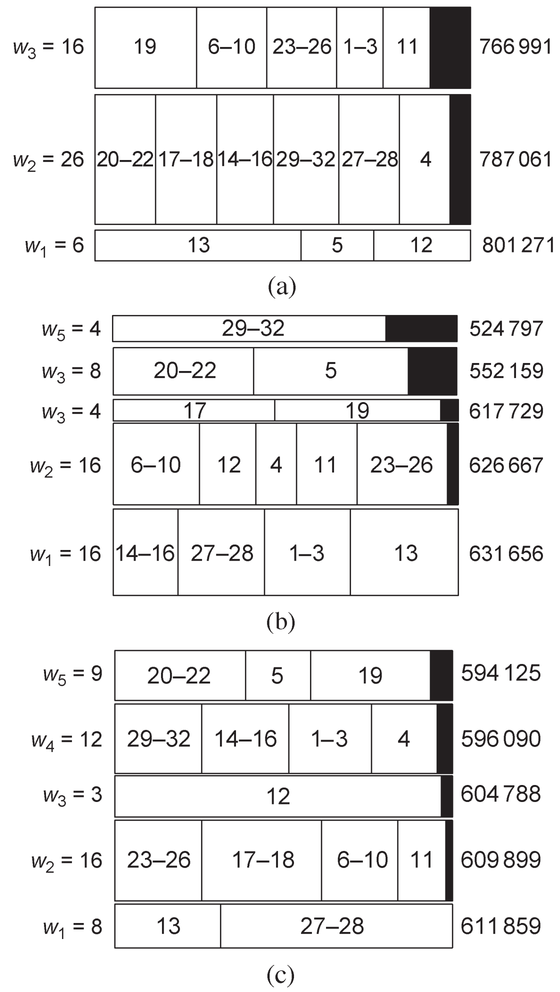


Fig. 8. Comparison of multilevel TAM design for SOC p93791 with $W_{\max} = 48$. (a) No bandwdth matching. (b) Type I converters. (c) Type II converters.

V. Experimental Results

To investigate the implication of the proposed solution on the tradeoff between test application time and DFT area overhead, benchmark SOCs from the ITC'02 SOC test benchmarking initiative [29], [30] are used in our experiment. We first show the savings in test application time with multifrequency TAM design in Section V-A for the flattened version of the benchmark SOCs. Next, we present the benefits of the proposed multilevel TAM design for four hierarchical benchmark SOCs.

A. Experiment 1: Test Application Time for Flattened SOCs

In this experiment, we have used the flattened versions, i.e., the hierarchy has not been taken into consideration, of the benchmark SOCs p22810, p34392, and p93791 from the ITC'02 benchmark suite [29]. For TestRail architecture, assuming fixed-length scan chains, we compare the test application time of mfTAMDesign with the serial schedule from TR-Architect [8]. For Test Bus architecture, assuming fixed-length scan chains, we compare the results of mfTAMDesign with six representative approaches: 1) the generalized rectangle packing (GRP) from [18]; 2) the simulated annealing algorithm from [40]; 3) ILP and exhaustive enumeration from [15]; 4) the heuristic Par eval from [16]; 5) the Lagrange multipliers from [35]; and 6) TR-Architect from [8]. The first two methods used flexible-width test architecture, while the others employed fixed-width test architecture. For flexible-length scan chains, we compare our results against [8]. Test application time is given in ATE clock cycles (note that when $f_{\text{tam}} \geq f_{\text{ext}}$ the result may need to be rounded up to an integer).

Tables I–III present experimental results for SOCs with fixed-length scan chains for the TestRail architecture, fixed-length scan chains for the Test Bus architecture and flexible-length scan chains, respectively. For each of the three SOCs, for a maximum value of TAM width W_{tam}, we show the test application time obtained by previous approaches and the test application time obtained by mfTAMDesign when: 1) all the internal TAMs run at the frequency $f_{\text{tam}} = f_{\text{ext}}$; 2) the TAMs can run at $f_{\text{tam}} \leq f_{\text{ext}}$; and 3) the TAMs can run at the frequency $f_{\text{tam}} \leq 2f_{\text{ext}}$ (note that TAMs can also run at a frequency lower than f_{ext} in this situation). We also give the virtual TAM width used inside the SOC when frequency conversion is used ($W_{\text{vt}-l}$

TABLE I
EXPERIMENTAL RESULTS FOR TESTRAIL ARCHITECTURE WITH FIXED-LENGTH SCAN CHAINS

SOC	$W_{\max}$	$f_{\mathrm{tam}} = f_{\mathrm{ext}}$		$f_{\mathrm{tam}} \leq f_{\mathrm{ext}}$		$f_{\mathrm{tam}} \leq 2f_{\mathrm{ext}}$	
		TR-Architect [8]	mfTAM	mfTAM$_l$	$W_{\mathrm{vt}-l}$	mfTAM$_h$	$W_{\mathrm{vt}-h}$
p22810	16	476 301	**454 443**	**437 976**	**23**	**437 078**	**11**
	24	310 249	310 249	**308 117**	**32**	**295 793**	**15**
	32	226 640	**223 466**	223 466	32	**218 988**	**23**
	40	190 995	190 995	**186 123**	**45**	**177 454**	**20**
	48	160 221	160 221	160 221	48	**154 059**	**32**
	56	145 417	145 417	145 417	56	**128 492**	**28**
	64	133 405	133 405	133 405	64	**111 733**	**32**
p34392	16	1 032 049	**1 010 821**	1 010 821	16	**975 968**	**12**
	24	721 244	**701 838**	**666 908**	**33**	**653 204**	**15**
	32	552 746	**551 249**	**544 579**	**33**	**505 411**	**16**
	40	544 579	544 579	544 579	33	**430 019**	**20**
	48	544 579	544 579	544 579	33	**333 454**	**33**
	56	544 579	544 579	544 579	33	**293 805**	**30**
	64	544 579	544 579	544 579	33	**272 290**	**33**
p93791	16	1 853 402	**1 824 376**	1 824 376	16	**1 783 085**	**8**
	24	1 240 305	**1 210 081**	1 210 081	24	**1 196 230**	**15**
	32	940 745	940 745	**906 878**	**37**	906 878	37
	40	786 608	**726 616**	**721 859**	**42**	721 859	42
	48	628 977	**611 730**	611 730	48	**605 041**	**24**
	56	530 059	530 059	530 059	56	**522 527**	**28**
	64	461 128	461 128	**455 738**	95	**453 439**	37

TABLE II
EXPERIMENTAL RESULTS FOR TEST BUS ARCHITECTURE WITH FIXED-LENGTH SCAN CHAINS

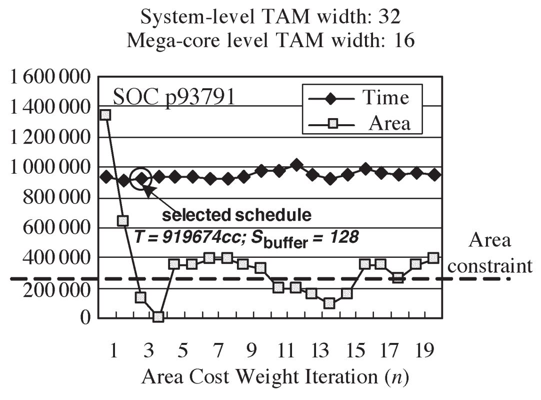
SOC	$W_{\max}$	$f_{\mathrm{tam}} = f_{\mathrm{ext}}$							$f_{\mathrm{tam}} \leq f_{\mathrm{ext}}$		$f_{\mathrm{tam}} \leq 2f_{\mathrm{ext}}$	
		Flexible width		**Fixed width**								
		GRP [18]	**SA_2** [40]	**ILP/enum** [15]	**Par eval** [16]	**Lagrange** [35]	**TR-Architect** [8]	mfTAM	mfTAM$_l$	$W_{\mathrm{vt}-l}$	mfTAM$_h$	$W_{\mathrm{vt}-h}$
p22810	16	489 192	438 619	462 210	468 011	434 922	458 068	**433 485**	**430 052**	**23**	**427 119**	**8**
	24	330 016	*293 019*	361 571	313 607	313 607	299 718	**296 365**	296 365	24	**287 327**	**15**
	32	245 718	219 923	312 659	246 332	245 622	222 471	**218 035**	218 035	32	**215 026**	**23**
	40	199 558	*180 004*	278 359	232 049	194 193	190 995	**183 550**	**182 678**	**47**	**176 810**	**20**
	48	173 705	*151 886*	278 359	232 049	164 755	160 221	**155 851**	155 851	48	**148 183**	**24**
	56	157 159	*132 812*	268 472	153 990	**145 417**	**145 417**	**145 417**	145 417	56	**126 487**	**28**
	64	142 342	*112 515*	260 638	153 990	133 628	**133 405**	**133 405**	**128 706**	**71**	**109 018**	**32**
p34392	16	1 053 491	*965 252*	**998 733**	1 033 210	1 021 510	1 010 821	1 006 155	1 006 155	16	**975 338**	**8**
	24	759 427	*657 561*	720 858	882 182	729 864	680 411	**663 193**	663 193	24	**646 382**	**15**
	32	544 579	544 579	591 027	663 193	630 934	551 778	**544 579**	544 579	32	**503 078**	**16**
	40	544 579	544 579	**544 579**	**544 579**	**544 579**	**544 579**	**544 579**	544 579	32	**392 441**	**20**
	48	544 579	544 579	**544 579**	**544 579**	**544 579**	**544 579**	**544 579**	544 579	32	**331 597**	**24**
	56	544 579	544 579	**544 579**	**544 579**	**544 579**	**544 579**	**544 579**	544 579	32	**292 262**	**30**
	64	544 579	544 579	**544 579**	**544 579**	**544 579**	**544 579**	**544 579**	544 579	32	**272 290**	**32**
p93791	16	1 932 331	*1 765 797*	**1 771 720**	1 786 200	1 775 586	1 791 638	1 791 638	1 791 638	16	**1 759 710**	**8**
	24	1 310 841	*1 178 397*	1 187 990	1 209 420	1 198 110	**1 185 434**	**1 185 434**	1 185 434	24	**1 177 054**	**12**
	32	988 039	893 892	**887 751**	894 342	936 081	912 233	906 878	902 716	42	895 819	16
	40	794 027	718 005	**698 583**	741 965	734 085	718 005	718 005	718 005	40	706 798	20
	48	669 196	*597 182*	**599 373**	**599 373**	**599 373**	601 450	601 450	601 450	48	**592 717**	**24**
	56	568 436	*510 516*	514 688	514 688	514 688	528 925	**513 564**	513 564	56	**511 158**	**28**
	64	517 958	*451 472*	460 328	473 997	472 388	**455 738**	**455 738**	455 738	64	**451 358**	**42**

and $W_{\mathrm{vt}-h}$). We select $\mathrm{layoutConstraint} = 1.5$, i.e., to keep routing overhead under control the internal virtual TAM wires should not exceed the tester channels by 50%.

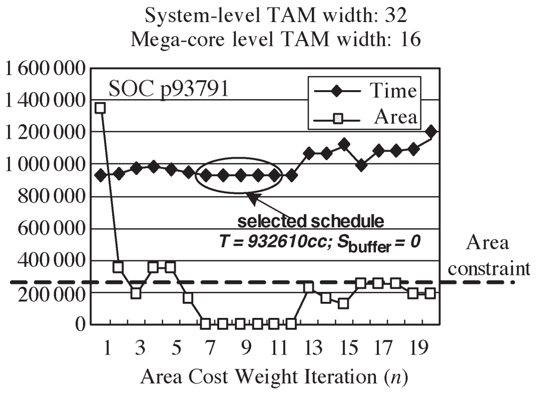
When $f_{\mathrm{tam}} = f_{\mathrm{ext}}$, no additional virtual TAMs are required, and hence, no additional hardware overhead is introduced. This means that although the algorithm starts the optimization with a large number of virtual TAM wires at a lower frequency, after merging multifrequency TAMs and distributing the freed virtual TAM wires, all the TAMs are finally designed to operate at the same frequency (f_{ext}). The test application time is reduced in several cases, which is due to a larger solution space exploration. This is achieved with a computation time of at most a few seconds, which is unlike the ILP method [15] that requires hours for large SOCs. It is important to mention that, in this case, since $f_{\mathrm{tam}} = f_{\mathrm{ext}}$ and no additional hardware needs to be introduced, the test application time can be improved with no penalty in area or power. It should also be noted that, although the proposed multifrequency TAM design algorithm builds on the search engine of TR-Architect [8], the basic principle used to expand the available TAM lines to a larger number of slower virtual TAMs and then merge them in order to explore a larger solution space is also applicable to other design

TABLE III
EXPERIMENTAL RESULTS FOR FLEXIBLE-LENGTH SCAN CHAINS

SOC	$W_{\max}$	Test Bus						TestRail					
		$f_{\text{tam}} = f_{\text{ext}}$		$f_{\text{tam}} \leq f_{\text{ext}}$		$f_{\text{tam}} \leq 2f_{\text{ext}}$		$f_{\text{tam}} = f_{\text{ext}}$		$f_{\text{tam}} \leq f_{\text{ext}}$		$f_{\text{tam}} \leq 2f_{\text{ext}}$	
		TR-Architect	mfTAM	mfTAM$_l$	$W_{\text{vt}-l}$	mfTAM$_h$	$W_{\text{vt}-h}$	TR-Architect	mfTAM	mfTAM$_l$	$W_{\text{vt}-l}$	mfTAM$_h$	$W_{\text{vt}-h}$
p22810	16	431 331	**428 303**	**427 466**	**19**	**424 493**	**8**	448 179	448 179	**436 281**	**20**	**431 242**	**11**
	32	218 982	**215 461**	**215 397**	**33**	**213 733**	**19**	223 368	**220 591**	**218 084**	**33**	218 084	33
	48	148 766	**144 809**	144 809	48	**143 002**	**24**	150 455	150 455	**146 706**	**70**	146 706	70
	64	110 922	**109 755**	109 755	64	**107 699**	**33**	112 695	**110 922**	**110 922**	**64**	**109 042**	**33**
p34392	16	971 816	**964 488**	**962 976**	**22**	**945 162**	**11**	1 003 345	1 003 345	1 003 345	16	**986 688**	**8**
	32	499 083	499 083	**483 402**	**43**	**481 488**	**22**	505 783	**490 246**	490 246	24	490 246	24
	48	326 400	326 400	**325 472**	**64**	**325 255**	**36**	347 948	347 948	**327 670**	**52**	**326 411**	**31**
	64	241 254	241 254	241 254	64	241 254	64	246 755	246 755	**245 776**	**95**	**245 123**	**32**
p93791	16	1 757 602	1 757 602	1 757 602	16	**1 755 868**	**8**	1 821 818	**1 819 559**	1 819 559	16	**1 783 085**	**8**
	32	885 071	**882 713**	882 713	32	**878 801**	**16**	946 311	**919 466**	**895 027**	**46**	895 027	46
	48	587 755	**587 571**	587 571	48	**586 631**	**24**	599 263	**596 224**	**595 703**	**69**	**593 039**	**27**
	64	444 195	**441 874**	**441 560**	**94**	**441 357**	**32**	451 094	451 094	**444 272**	**75**	444 272	75



(a)



(b)

Fig. 9. Test application time and DFT area for p93791 with different area cost weights. (a) $p = 50$; (b) $p = 250$.

space exploration strategies for fixed-width TAM architectures. When $f_{\text{tam}} = f_{\text{ext}}$, as it can be observed in Table II, the results from [40] are the only ones that are slightly better than the proposed solution in several cases. However, this approach uses a flexible-width test architecture, which cannot be easily extended to the proposed modular multifrequency TAM architecture, whose benefits in further reducing test application time, when f_{ext} is lower or higher than f_{tam}, are discussed next.

When $f_{\text{tam}} \leq f_{\text{ext}}$, the result is improved in several cases, however, at the expense of additional virtual TAM wires inside the SOC $(W_{\text{vt}-l} > W_{\max})$. When $f_{\text{tam}} \leq 2f_{\text{ext}}$, it can be seen in most cases that a lower test application time can be achieved with a small number of virtual TAM wires $(W_{\text{vt}-h} < W_{\max})$ running at a frequency higher than f_{ext}. As long as power ratings allow it, this technique is particularly attractive when one core becomes the bottleneck and increasing TAM width will not result in any improvement. For example, for SOC p34392, assuming fixed-length scan chains, when $W_{\text{tam}} \geq 32$ (for Test Bus architecture) or $W_{\text{tam}} \geq 33$ (for TestRail architecture), the test application time does not decrease any more. If the additional TAM wires are used to combine to a lower number of virtual TAM wires running at a higher frequency (within the power rating of the SOC), the test application time can be significantly decreased for TAM widths from 33 to 64. As shown in Tables I–III, significant savings are achieved only for the case when the TAM frequency is greater than the ATE frequency $(f_{\text{tam}} \leq 2f_{\text{ext}})$. However, it is important to note that there are several examples where the results for $f_{\text{tam}} \leq f_{\text{ext}}$ and $f_{\text{tam}} = f_{\text{ext}}$ show an advance over the current methods (especially when considering the fact that, given the same constraints, the experimental results do not vary significantly for the solutions reported in Tables I–III).

B. Experiment 2: Test Application Time for Hierarchical SOCs

In this section, we show the advantages of using frequency converters for hierarchical SOCs with hard megacores. Before comparing the proposed multilevel TAM design method against the only existing approach [14] that tackled the same problem, we illustrate the importance of considering the DFT area overhead in the cost of function used for guiding the design space exploration. Fig. 9 shows the variation of test application time and DFT area, caused by frequency converters, for different area cost weights $\text{costWeight} = n \times p$. Based on the value of the scaling factor p, the area cost weight varies from 0 to 950 with an interval value of 50 in Fig. 9(a); while in Fig. 9(b), it varies from 0 to 4750 with an interval value of 250. The area is scaled as $S_{\text{buffer}} \times 1000$ to be seen clearly in the figure. We can observe that when area cost is not taken into account $(n = 0)$, the added DFT area for frequency converters is quite large, and hence, this solution is not preferable. We can also observe an increase in SOC testing time when the area cost weight is high

TABLE IV
TEST APPLICATION TIME OF MULTILEVEL DESIGN FOR HIERARCHICAL SOCs p22810 AND a586710

SOC	W_{tam}	T [14]	T_ϕ	ΔT_ϕ (percent)	T_{I} ($S_{\text{buffer}} \leq 50$)	S_{buffer}	ΔT_{I} (percent)	T_{II} ($S_{\text{buffer}} \leq 500$)	S_{buffer}	ΔT_{II} (percent)
p22810	8	-	992 297	-	890 905	32	-10.22	870 622	128	-12.26
	16	505 858	496 804	-1.79	474 023	16	-4.59	446 383	96	-10.15
	24	412 682	353 619	-14.31	322 370	48	-8.84	317 063	96	-10.34
	32	396 473	280 634	-29.22	280 634	0	0	280 634	0	0
	40	366 260	280 634	-23.38	280 634	0	0	280 634	0	0
	48	366 260	280 634	-23.38	280 634	0	0	280 634	0	0
	64	366 260	280 634	-23.38	280 634	0	0	280 634	0	0
a586710	8	-	7.95×10^7	-	7.95×10^7	0	0	7.95×10^7	0	0
	16	-	5.27×10^7	-	4.32×10^7	16	-18.03	4.21×10^7	80	-20.11
	24	3.06×10^7	3.06×10^7	0	2.87×10^7	16	-6.21	2.87×10^7	16	-6.21
	32	2.88×10^7	2.19×10^7	-23.96	2.19×10^7	0	0	2.19×10^7	0	0
	40	2.50×10^7	1.91×10^7	-23.60	1.91×10^7	0	0	1.91×10^7	0	0
	48	2.14×10^7	1.53×10^7	-28.50	1.53×10^7	0	0	1.53×10^7	0	0
	64	2.14×10^7	1.41×10^7	-34.11	1.41×10^7	0	0	1.41×10^7	0	0

TABLE V
TEST APPLICATION TIME OF MULTILEVEL TAM DESIGN FOR HIERARCHICAL SOCs p34392 AND p93791

SOC	W_{tam}	T [14]	T_ϕ	ΔT_ϕ (percent)	T_{I} ($S_{\text{buffer}} \leq 100$)	S_{buffer}	ΔT_{I} (percent)	T_{II} ($S_{\text{buffer}} \leq 1000$)	S_{buffer}	ΔT_{II} (percent)
p34392	8	x	x	x	2 281 733	96	x	2 224 022	416	x
	16	-	1 467 705	-	1 226 208	96	-16.45	1 156 969	672	-21.17
	24	1 347 023	1 467 705	+8.96	834 337	96	-43.15	781 906	608	-46.73
	32	788 873	776 537	-1.56	691 168	96	-10.99	626 821	736	-19.28
	40	728 426	776 537	+6.60	606 261	32	-21.93	606 261	32	-21.93
	48	618 597	606 261	-1.99	606 261	0	0	606 261	0	0
	64	618 597	606 261	-1.99	606 261	0	0	606 261	0	0
p93791	8	x	x	x	3 697 813	64	x	3 701 007	576	x
	16	2 044 124	1 865 140	-8.76	1 865 140	0	0	1 804 088	256	-3.27
	24	1 351 710	1 486 628	+9.98	1 240 103	96	-16.58	1 198 247	768	-19.40
	32	1 087 300	1 486 628	+36.73	921 805	96	-37.99	913 078	640	-38.58
	40	839 796	801 271	-4.59	756 160	64	-5.63	729 651	896	-8.94
	48	839 796	801 271	-4.59	631 656	64	-21.17	611 859	608	-23.64
	64	839 796	553 775	-34.06	473 503	0	-14.50	473 503	0	-14.50

[$n \geq 12$ in Fig. 9(b)], which is not desirable either. Therefore, the introduction of the area cost into the scheduling algorithm helps the system integrator to keep the added area within predefined limits, while still minimizing the test application time (whose variation is not as large as the variation in the DFT area). However, due to the nature of the heuristic approach, the SOC testing time is not monotonically increasing and the total DFT area cost is not monotonically decreasing with the increase of the area cost weight. Nevertheless, if the system integrator provides an area constraint (shown in the figure using the dotted line), the proposed Algorithm 4 from Section IV-B will select the solution with the minimum test application time that fulfills the given area constraint. In this experiment for SOC p93791, when the scaling factor is $p = 50$, we obtain a better result in terms of test application time when compared to $p = 250$.

In Tables IV and V, we compare the results of our solution with the results from [14] for noninteractive design transfer model. The reason we consider only the noninteractive model is that the interactive model assumes that TAM design for megacores is flexible, which conflicts with the objective to reuse hard megacores. Our results indicate that as long as the system integrator accepts additional test area, savings can be achieved in test application time without any internal modification to the hard megacores. In this experiment, core-level TAM widths supplied to each megacore before system-level TAM design have the same configurations as in [14]. That is, in Table IV, for p22810 and a586710, we have $W_l = 8$ bits, while in Table V, for p34392 and p93791, we have $W_l = 16$ bits. In addition, the scaling factor is $p = 50$. T, T_ϕ, T_{I}, and T_{II} denote the test application time from [14], the test application time obtained by using the mfTAMDesign algorithm, the test application time obtained by Algorithm 4 from Section IV-B when the area constraint for frequency converters is stringent (50 for p22810 and a587610; 100 for p34392 and p93791), and when the area constraint for frequency converters is relaxed (500 for p22810 and a587610; 1000 for p34392 and p93791), respectively. The percentage change is calculated as $\Delta T_\phi = (T_\phi - T)/T$, $\Delta T_{\text{I}} = (T_{\text{I}} - T_\phi)/T_\phi$, and $\Delta T_{\text{II}} = (T_{\text{II}} - T_\phi)/T_\phi$, respectively. To provide a fair comparison and prove the benefits caused exclusively by the use of Algorithm 4 (which is orthogonal to any single-level TAM design algorithm), we report ΔT_{I} and ΔT_{II} with respect to T_ϕ instead of with respect to T [14].

For T_ϕ, we have the same hierarchical design flow as in [14]. It can be seen in most of the cases that $T_\phi < T$, however, because the TAM width constraint for megacores restricts the CreateStartSolution step in TR-Architect [8] (used in mfTAMDesign); in a few cases, the algorithm from [14] gives better results. It is worth noting that both methods cannot provide a solution for SOC p34392 and p93791 when the system-level TAM width is only 8 because the system-level TAM is not wide enough to fork out to the core-level TAM ($W_l = 16$). By

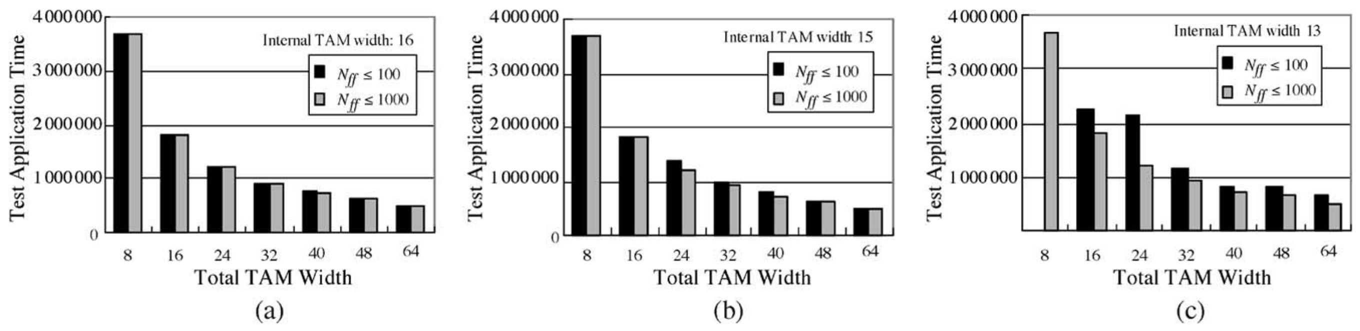


Fig. 10. Test application time for SOC p93791 with different internal TAM widths. (a) Internal TAM width 16; (b) internal TAM width 15; (c) internal TAM width 13.

introducing the frequency converter next to the wrapper of the megacore, the system integrator can afford narrow system-level TAM design. In almost all cases, the test application time is decreased when frequency converters can be used to map system-level TAM to core-level TAM, and, when the area constraint is relaxed, the test application time is further decreased (i.e., $T_{\mathrm{II}} \leq T_{\mathrm{I}} \leq T_{\phi}$). This improvement is due to the greater flexibility (the system integrator has more choices for the system-level TAM width assignment to the megacore) during test scheduling. It is important to note that for SOC p93791, when $W_{\mathrm{tam}} = 64$, $T_{\mathrm{I}} = T_{\mathrm{II}} < T_{\phi}$ can be acquired without introducing converters. This is also because the algorithm searches a larger solution space and happens to grant all the megacores 16 TAM lines. Note that despite exploring a larger solution space, the computation time is still at most within seconds, which demonstrates that the proposed solution will not have any impact on test development time. For SOC p22810, when $W_{\max} \geq 32$, and for SOC p34392, when $W_{\max} \geq 40$, one of the megacores inside the SOC becomes the bottleneck TAM on its own (megacore 1 for p22810 and megacore 18 for p34392). Since the system-level TAM width assigned to the megacore already exceeds the internal core-level TAM width, the test application time for the entire SOC can no longer be decreased. As a result, a total system-level TAM width of 32 for p22810 and 40 for p34392 would be an effective choice for the system integrator.

Fig. 10 shows the influence of the internal TAM width of the megacores on the test schedule. In the three subfigures, the internal TAM widths of all the megacores in SOC p93791 are set as 16, 15, and 13, respectively. When the area constraint is set to 1000, type II converters can be used and each megacore in all cases can be assigned to any arbitrary TAM width. When the area constraint is set as 100, however, because of the large size of type II converter, only the type I converter is available for matching the TAM widths. When the internal TAM width is 16, we have five choices for the TAM width assigned to the megacores (1, 2, 4, 8, and 16). When the internal TAM width is 15, we have four choices for the TAM width assigned to the megacores (1, 3, 5, and 15). When the internal TAM width is 13, we only have two choices for the TAM width assigned to the megacores (1 and 13). With the decreasing number of options for TAM widths, the solution space that the scheduling algorithm can exploit is smaller, and hence, it leads to solutions with longer test application time. As it can be seen in Fig. 10, the difference between the two area constraints is much larger when the internal TAM width is set as 13. It should also be noted that we cannot even find a solution to fulfill the constraint $S_{\mathrm{buffer}} \leq 100$ when the total TAM width is 8.

VI. Conclusion

This paper has introduced a new method, based on the bandwidth matching technique, for designing multifrequency TAMs for modular hierarchical SOC testing. Using experimental results, it was shown that with limited area overhead, we can reuse hard megacores and achieve reduced test application time compared to prior work. The design space exploration framework described in this paper can be used to rapidly explore the test application time/area overhead tradeoffs for hierarchical SOCs and recommend the cost-effective solutions to the system integrator.

Acknowledgment

The authors wish to acknowledge the anonymous reviewers for their constructive comments.

References

[1] J. Aerts and E. J. Marinissen, "Scan chain design for test time reduction in core-based ICs," in *Proc. IEEE Int. Test Conf. (ITC)*, Washington, DC, Oct. 1998, pp. 448–457.

[2] M. Benabdenbi, W. Maroufi, and M. Marzouki, "CAS-BUS: A test access mechanism and a toolbox environment for core-based system chip testing," *J. Electron. Test., Theory Appl.*, vol. 18, no. 4/5, pp. 455–473, Aug. 2002.

[3] A. Benso, S. D. Carlo, P. Prinetto, and Y. Zorian, "A hierarchical infrastructure for SOC test management," *IEEE Des. Test. Comput.*, vol. 20, no. 4, pp. 32–39, Jul. 2003.

[4] K. Chakrabarty, "Optimal test access architectures for system-on-a-chip," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 6, no. 1, pp. 26–49, Jan. 2001.

[5] S. Dutta, R. Jensen, and A. Rieckmann, "Viper: A multiprocessor SOC for advanced set-top box and digital TV systems," *IEEE Des. Test. Comput.*, vol. 18, no. 5, pp. 21–31, Sep.–Oct. 2001.

[6] S. K. Goel, "An improved wrapper architecture for parallel testing of hierarchical cores," in *Proc. IEEE Eur. Test Symp. (ETS)*, Corsica, France, May 2004, pp. 147–152.

[7] S. K. Goel, K. Chiu, E. J. Marinissen, T. Nguyen, and S. Oostdijk, "Test infrastructure design for the Nexperia home platform PNX8550 system chip," in *Proc. Design, Automation, and Test in Europe (DATE) Designers Forum*, Paris, France, Feb. 2004, pp. 108–113.

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

[9] ——, "Control-aware test architecture design for modular SOC testing," in *Proc. IEEE Eur. Test Workshop (ETW)*, Maastricht, The Netherlands, May 2003, pp. 57–62.

[10] ——, "Layout-driven SOC test architecture design for test time and wire length minimization," in *Proc. Design, Automation, and Test in Europe (DATE)*, Munich, Germany, Mar. 2003, pp. 738–743.

[11] D. Heidel, S. Dhong, P. Hofstee, M. Immediato, K. Nowka, J. Silberman, and K. Stawiasz, "High speed serializing/de-serializing design-for-test method for evaluating a 1 GHz microprocessor," in *Proc. IEEE VLSI Test Symp. (VTS)*, Princeton, NJ, 1998, pp. 234–238.

[12] Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, O. Samman, Y. Zaidan, and S. M. Reddy, "Resource allocation and test scheduling for concurrent test of core-based SOC design," in *Proc. IEEE Asian Test Symp. (ATS)*, Kyoto, Japan, Nov. 2001, pp. 265–270.

[13] International SEMATECH. (2001). *The International Technology Roadmap for Semiconductors (ITRS): 2001 Edition*. [Online]. Available: http://public.itrs.net/Files/2001ITRS/Home.htm

[14] V. Iyengar, K. Chakrabarty, M. D. Krasniewski, and G. N. Kumar, "Design and optimization of multilevel TAM architectures for hierarchical SOCs," in *Proc. IEEE VLSI Test Symp. (VTS)*, Napa, CA, Apr. 2003, pp. 299–304.

[15] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Cooptimization of test wrapper and test access architecture for embedded cores," *J. Electron. Test., Theory Appl.*, vol. 18, no. 2, pp. 213–230, Apr. 2002.

[16] ——, "Efficient wrapper/TAM cooptimization for large SOCs," in *Proc. Design, Automation, and Test in Europe (DATE)*, Paris, France, Mar. 2002, pp. 491–498.

[17] ——, "Integrated wrapper/TAM cooptimization, constraint-driven test scheduling, and tester data volume reduction for SOCs," in *Proc. ACM/IEEE Design Automation Conf. (DAC)*, New Orleans, LA, Jun. 2002, pp. 685–690.

[18] ——, "On using rectangle packing for SOC wrapper/TAM cooptimization," in *Proc. IEEE VLSI Test Symp. (VTS)*, Monterey, CA, Apr. 2002, pp. 253–258.

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

[20] V. Iyengar, S. K. Goel, K. Chakrabarty, and E. J. Marinissen, "Test resource optimization for multisite testing of SOCs under ATE memory depth constraints," in *Proc. IEEE Int. Test Conf. (ITC)*, Baltimore, MD, Oct. 2002, pp. 1159–1168.

[21] A. Khoche, "Test resource partitioning for scan architectures using bandwidth matching," in *Dig. Int. Workshop Test Resource Partitioning*, Baltimore, MD, 2002, pp. 1.4.1–1.4.8.

[22] S. Koranne, "A novel reconfigurable wrapper for testing of embedded core-based SOCs and its associated scheduling algorithm," *J. Electron. Test., Theory Appl.*, vol. 18, no. 4/5, pp. 415–434, Aug. 2002.

[23] ——, "Formulating SOC test scheduling as a network transportation problem," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 21, no. 12, pp. 1517–1525, Dec. 2002.

[24] ——, "On test scheduling for core-based SOCs," in *Proc. Int. Conf. Very Large Scale Integration Design*, Bangalore, India, Jan. 2002, pp. 505–510.

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

[26] J.-F. Li, H.-J. Huang, J.-B. Chen, C.-P. Su, C.-W. Wu, C. Cheng, S.-I. Chen, C.-Y. Hwang, and H.-P. Lin, "A hierarchical test scheme for system-on-chip designs," in *Proc. Design, Automation, and Test Europe (DATE)*, Munich, Germany, Feb. 2002, pp. 486–490.

[27] E. J. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousberg, and C. Wouters, "A structured and scalable mechanism for test access to embedded reusable cores," in *Proc. IEEE Int. Test Conf. (ITC)*, Washington, DC, Oct. 1998, pp. 284–293.

[28] E. J. Marinissen, S. K. Goel, and M. Lousberg, "Wrapper design for embedded core test," in *Proc. IEEE Int. Test Conf. (ITC)*, Atlantic City, NJ, Oct. 2000, pp. 911–920.

[29] E. J. Marinissen, V. Iyengar, and K. Chakrabarty. *ITC'02 SOC Test Benchmarks Web Site*. [Online]. Available: http://www.extra.research.philips.com/itc02socbenchm/

[30] ——, "A set of benchmarks for modular testing of SOCs," in *Proc. IEEE Int. Test Conf. (ITC)*, Baltimore, MD, Oct. 2002, pp. 519–528.

[31] E. J. Marinissen, R. Kapur, M. Lousberg, T. Mclaurin, M. Ricchetti, and Y. Zorian, "On IEEE P1500's standard for embedded core test," *J. Electron. Test., Theory Appl.*, vol. 18, no. 4/5, pp. 365–383, Aug. 2002.

[32] A. P. Niranjan and P. Wiscombe, "Islands of synchronicity, a design methodology for SOC design," in *Proc. Design, Automation, and Test Europe (DATE)*, Paris, France, 2004, pp. 64–69.

[33] A. Sehgal and K. Chakrabarty, "Efficient modular testing of SOCs using dual-speed TAM architectures," in *Proc. Design, Automation, and Test Europe (DATE)*, Paris, France, Feb. 2004, pp. 422–427.

[34] A. Sehgal, S. K. Goel, E. J. Marinissen, and K. Chakrabarty, "IEEE P1500-compliant test wrapper design for hierarchical cores," in *Proc. IEEE Int. Test Conf. (ITC)*, Charlotte, NC, Oct. 2004, pp. 1203–1212.

[35] A. Sehgal, V. Iyengar, M. D. Krasniewski, and K. Chakrabarty, "Test cost reduction for SOCs using virtual TAMs and Lagrange multipliers," in *Proc. ACM/IEEE Design Automation Conf. (DAC)*, Anaheim, CA, Jun. 2003, pp. 738–743.

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

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

[38] D. M. Wu, M. Lin, S. Mitra, K. S. Kim, A. Sabbavarapu, T. Jaber, P. Johnson, D. March, and G. Parrish, "H-DFT: A hybrid DFT architecture for low-cost high quality structural testing," in *Proc. IEEE Int. Test Conf. (ITC)*, Charlotte, NC, Oct. 2003, pp. 1229–1238.

[39] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing embedded-core-based system chips," *IEEE Computer*, vol. 32, no. 6, pp. 52–60, Jun. 1999.

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

Qiang Xu (S'03) received the B.E. and M.E. degrees in telecommunication engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 1997 and 2000, respectively, and the Ph.D. degree in electrical and computer engineering from McMaster University, Hamilton, ON, Canada, in 2005.

He is an Assistant Professor of Computer Science and Engineering at The Chinese University of Hong Kong. His research interest lies in the broad area of computer-aided design with special emphasis on test and debug of system-on-a-chip integrated circuits.

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

Nicola Nicolici (S'00–M'00) received the Dipl. Ing. degree in computer engineering from the University of Timisoara, Timisoara, Romania, in 1997, and the Ph.D. degree in electronics and computer science from the University of Southampton, Southampton, U.K., in 2000.

He is an Assistant Professor of Computer Engineering at McMaster University, Hamilton, ON, Canada. His research interests are in the area of computer-aided design and test. He has authored a number of papers in this area.

Dr. Nicolici received the IEEE TTTC Beausang Award for the Best Student Paper at the International Test Conference (ITC 2000) and the Best Paper Award at the IEEE/ACM Design Automation and Test in Europe Conference (DATE 2004). He is a member of the ACM SIGDA and the IEEE Computer and Circuits and Systems Societies and he serves on the editorial board of IEE Proceedings—Computers and Digital Techniques.