A Compression-Driven Test Access Mechanism Design Approach*

Paul Theo Gonciari and Bashir M Al-Hashimi
School of ECS, University of Southampton
United Kingdom
{ptg,bmah}@ecs.soton.ac.uk

Abstract

Driven by the industrial need for low-cost test methodologies, the academic community and the industry alike have put forth a number of efficient test data compression (TDC) methods. In addition, the need for core-based System-on-a-Chip (SoC) test led to considerable research in test access mechanism (TAM) design. While most previous work has considered TAM design and TDC independently, this work analyzes the interrelations between the two, outlining that a minimum test time solution obtained using TAM design will not necessarily correspond to a minimum test time solution when compression is applied. This is due to the dependency of some TDC methods on test bus width and care bit density, both of which are related to test time, and hence to TAM design. Therefore, this paper illustrates the importance of considering the characteristics of the compression method when performing TAM design, and it also shows how an existing TAM design method can be enhanced toward a compression-driven solution.

1. Introduction

In recent years the semiconductor industry has seen a significant increase in manufacturing test cost [13] due to high volume of test data [17], large test application times [3], insufficient channel capacity [19] and high cost of automatic test equipment (ATE) [13]. To alleviate the cost problem numerous approaches, mainly targeting test data compression, have been proposed. The main thrust behind test data compression (TDC) is that reducing the amount of test data and test time required to test a chip will reduce the load on the tester and hence reduce cost. This has been accomplished by compressing the test data using a compression algorithm, and introducing a decoder or a decompresser on-chip, which will expand the compressed test data into the initial test data. TDC has been approached from a number of research directions. For example, methods which **(i)** exploit the sparseness of care bits in the test set have been proposed in [2, 6, 17], and methods which **(ii)** exploit the regularities within the test set have been proposed in [5, 12, 18, 21].

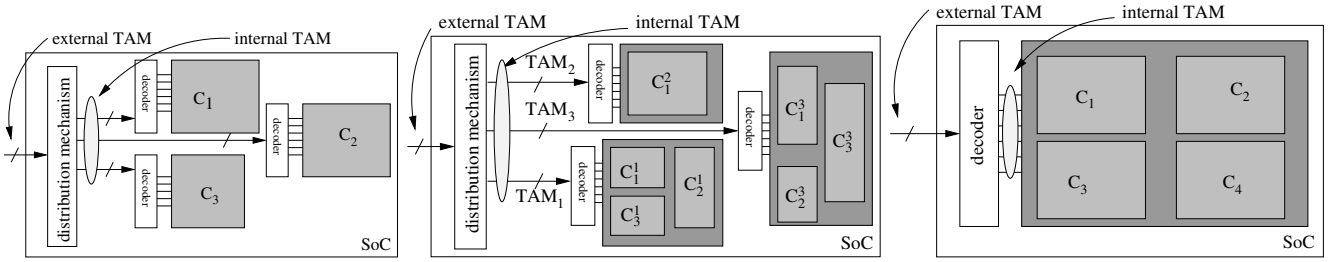
In addition to the high cost of test, high integration facilitates the manufacturing of entire systems-on-a-chip (SoC); a paradigm which brings forth new issues in testing [24], such as limited knowledge of the cores and restricted access to the cores. To cope with these restrictions test infrastructures have been developed to ensure proper test of the embedded cores. The advocated SoC test infrastructure is based on a core wrapper (e.g., TestShell [7], IEEE P1500 [23]) – to provide the environment required to test a core and its surroundings – and a test access mechanism (TAM) (e.g., test bus, TestRail [7]) – to transport stimuli to/from the embedded core to the test inputs/outputs of the SoC. Understanding and providing solutions for SoC test led to considerable research into core wrapper and TAM design. These problems have been tackled for different constraints (e.g., test time, test bus width, power dissipation, hierarchical information, control overhead, routing and layout) using various heuristics [8, 15, 20, 22].

This paper presents a test solution in which TAM design and TDC are combined into a unified problem formulation under core based SoC constraints – i.e., the system integrator has only the mandatory test information specified with a core as required by P1500 [23], being able to perform core wrapper design and has the test set delivered with the core. Also, he/she may be restrained from performing fault simulation due to IP protection of hard cores. We focus on TDC methods which exploit the spareness of care bits. Firstly, we analyze possible interactions between TDC and TAM in Section 2; and motivate the need for the TDC-TAM integration in the case of test bus width **sensitive** compression methods in Section 3. To facilitate this integration, in Section 4 a test time estimation function for the compression method considered in this paper is given; and we extend an existing TAM design method to incorporate the TDC method characteristics, in Section 5. Section 6 provides experimental results, and Section 7 concludes the paper.

2. Analysis of TDC-TAM interaction

In this section we provide an analysis of previous work which focuses on integrated TDC–TAM test solutions [5, 9, 10, 16], considering a control and area-overhead perspective. Previous work analysis for TDC and TAM design are detailed in [12, 17] and [8], respectively.

*This research is supported by EPSRC (UK) grant GR/S41135.



(a) *Core level*–TDC scenario (b) *TAM level*–TDC scenario (c) *TAM add-on*–TDC scenario

Figure 1. Possible TDC–SoC test scenarios

In addition to the area penalty involved with TDC, inserting TDC in a design also requires an increase in control overhead. Also, in TDC the number of test bus lines driving the on-chip decoder, and the number of decoder's outputs driving the core under test may be unequal. Therefore, when considering integrated TDC–SoC test, the question is: "Where will TDC interact with the existing SoC test infrastructure ?". We distinguish three scenarios as illustrated in Figure 1: **(a)** *core level*–TDC scenario (Figure 1(a)); **(b)** *TAM level*–TDC scenario (Figure 1(b)); and **(c)** *TAM add-on*–TDC scenario (Figure 1(c)).

Common to the three scenarios, in Figure 1, is that the TAM, which feeds the system is not necessarily the TAM which feeds the on-chip decoders (see Figures 1(a) and 1(b)) or the cores (see Figure 1(c)). In the figures we denote the TAM which feeds the SoC as the *external TAM*, and the TAM which feeds the internal cores, or the decoders, as the *internal TAM*. Corresponding to the external TAM we introduce the *external test time* ($time_{ext}$) and the *external test data* ($data_{ext}$), i.e., the time required to test the SoC and the corresponding amount of test data; and with respect to the internal TAM we introduce *internal test time* ($time_{int}$) and *internal test data* ($data_{int}$), i.e., the amount of test time and test data without TDC. The difference between the two types of TAM will become more apparent in the following paragraphs where we detail the three scenarios.

Core level–TDC In the first scenario **(a)**, the system integrator chooses to provide for each core its own decoder, decoders which are then connected to a distribution mechanism. This mechanism can be an interleaving architecture as illustrated in [5], a distributed architecture as proposed in [10], or a time multiplexing scheme [16]. In all the cases [5, 10, 16] the distribution mechanism has the external TAM of width much smaller than the internal TAM; for example $1 : n$ (for n cores) in [5, 10]. However, since each core is provided with a decoder, for a large number of cores the area and control overhead is considerable.

TAM level–TDC In the second scenario **(b)**, the system integrator chooses to provide a group of cores with one decoder. Similar to scenario **(a)** the decoders are then connected to a distribution mechanism. In this case however, the distribution mechanism can be a simple buffer [16], when the internal TAM and the external TAM are the same, or a time multiplexing scheme [9]. As noted in [16] the decoder and control overhead of this scheme is reduced in comparison with scenario **(a)**, at the expense of increased test time. While avoided in the figure for clarity the number of decoder's outputs which drive the groups of cores may not be equal among all groups.

TAM add-on–TDC In the third scenario **(c)**, the system integrator chooses to perform a TAM design for the entire system, and then provide *one* decoder for decompression. This scenario has been analyzed in [9] where the extended distribution architecture has been introduced. The scheme presented in [9] has the advantage of reduced control over **(a)** and **(b)**, at the expense of small test time penalties.

The approaches analyzed above [5, 9, 10, 16] use TDC methods which are either single-scan chain based [5, 10, 16] or are not sensitive to the test bus width [9]. Therefore, these compression methods can be added on-top of existing TAM design solutions without any changes to the design flow, and with small performance degradation, in terms of test time and test data, of the compression method. This has been illustrated in [9] and [11] where the test data and test time attained after compression appeared to be invariant to bus width changes. In this paper, however, we address the problem of integrating in TAM design a TDC method which is **sensitive** to the test bus width. Hence, performing compression on-top of an existing TAM solution may lead to inefficient test bus usage and considerable performance degradation with respect to test time. This aspect is analyzed in this paper for scenario **(c)**.

3. Preliminaries and motivation

As noted in Section 1, the core based SoC paradigm may prohibit the usage of fault simulation during test preparation. Therefore, the system integrator is restrained from using TDC methods which may require automated test pattern generation (ATPG) integration. For these reasons, in [11] the $XOR-Network$ approach from [2] has been tailored for core based SoC test. In the remainder of this paper we will use the $XOR-Network$ method as introduced in [11], and, in order to avoid confusion, we will refer to this particular implementation as $XNet$. It should be noted that while we illustrate the TDC-TAM integration problem for $XNet$, the mechanism described in this paper is generally applicable to similar types of compression methods.

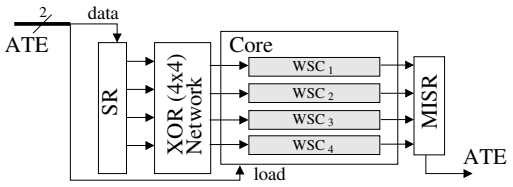


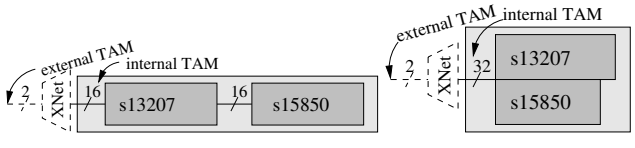
Figure 2. $XNet$ based on the architecture from [2]

XNet architecture is illustrated in Figure 2 for a core with 4 wrapper scan chains (WSCs). For a given test cube, the main idea behind the method is to stream data into the shift register (*SR*), which will then justify through the XOR network the care bits into the WSCs. However, if the number of care bits is too high, the architecture may run into temporal pattern lockout (i.e., the inability to justify the care bits at a given moment), and to account for this case a control signal to halt the WSC load temporarily is added. Therefore, the architecture requires two ATE channels: one to feed data and one to control the load. And hence, the number of bits representing test data will be double the number of clock cycles representing test time. Further details about the implementation of this approach can be found in [11]. Throughout the paper we denote $XNet(w)$ the case when XNet is applied to a test bus width of w.

The fact that $XNet$ is sensitive to w has been pointed out in [11], where it has been shown that, for the same test set, the number of temporal pattern lockouts increases with the increase in w. The following example illustrates the implications of this fact in TAM design.

Example 1 Consider in Figure 3(a) a SoC composed of 2 cores, from the ISCAS89 benchmark suite [4], assigned to a test bus of width 16. The test sets used for the two cores have the following parameters (test data size in bits, and test time in ATE clock cycles): $data_{s13207} = 243200$ and $time^{16}_{s13207} = 15200$; and $data_{s15850} = 209840$ and $time^{16}_{s15850} = 13115$. Without compression, the internal test time and test data of the system is the addition of the individual's cores test data and test time, respectively, resulting in $time_{int} = 28315$, and $data_{int} = 453040$. Applying the $XNet(16)$, the two cores have a test time of $time^{c}_{s13207} = 22671$ and $time^{c}_{s15850} = 25672$, respectively. Hence, $time_{ext} = time^{c}_{s13207} + time^{c}_{s15850} = 48343$ and $data_{ext} = 2 * time_{ext} = 96686$ which is $\approx 5x$ less than $data_{int}$. Testing the two cores in parallel, using a 32 bit internal TAM (see Figure 3(b)), the $time_{int} = max\{15200, 13115\} = 15200$, which brings a 46% reduction in test time. When we apply $XNet(32)$ to the compound test set, the external test time becomes $time_{ext} = time^{c}_{s13207+s15850} = 41554$, which represents only a 14% reduction in external test time. The external test data is $data_{ext} = 2 * time_{ext} = 83108$.

In summary, there are two observations that can be drawn from Example 1. Firstly, doubling the test bus width toward obtaining an ideal 50% reduction in $time_{int}$ will not be followed by a similar reduction in $time_{ext}$. And secondly, obtaining the external test time requires the application of the compression method after each new TAM assignment.



(a) $w = 16$ (b) $w = 32$

Figure 3. Example

A direct implication of the first observation is the choice of Pareto-points used in a number of TAM design algorithms (e.g., [15]). We note that the $time_{ext}$ to w relation corresponding to Pareto-points does not exhibit the same behavior as the original $time_{int}$ to w relation. For example, for core s38584, the Pareto-points ($time_{int}$) and the corresponding external test times ($time_{ext}$) are shown in Figure 4(a). Note that not only the Pareto-points are different, but also that increasing w does not always lead to a reduction in test time, e.g., $w = \overline{8 \ldots 12}$. Hence, in compression-driven TAM design care must be taken to avoid using certain Pareto-points since they may lead to solutions with worse $time_{ext}$. Therefore, if compression-unaware TAM design is performed the obtained design may exhibit inefficient test bus usage and test time penalties.

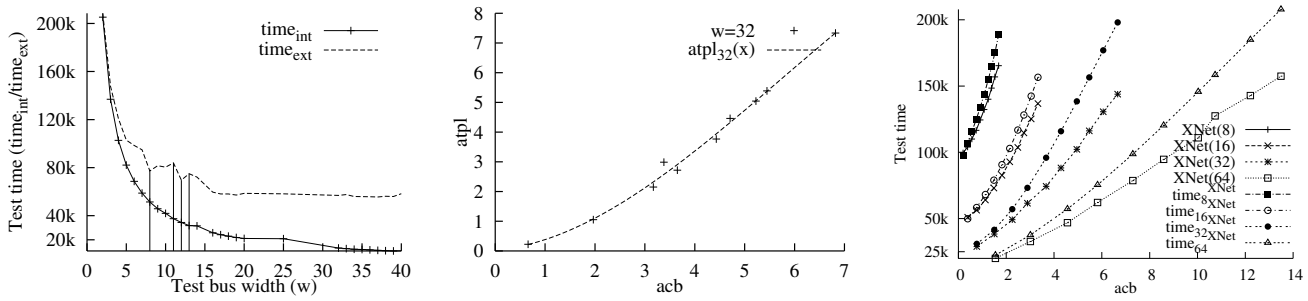
An implication of the second observation relates to the computational time required to perform a TDC-TAM design. This is because, the external test time does not only depend on the core's external times, but also on core's relative position in the TAM assignment, and hence the compression method must be applied after each new TAM assignment. With reference to Example 1, for $w = 32$, the test time is not merely a simple $max\{time^{c}_{s13207}, time^{c}_{s15850}\}$ as in the case of TAM design, instead $time_{ext}$ has been computed by applying $XNet(32)$ to the compound test set. Since TDC is applied to systems with large volumes of test data, which will take a long time to compress, the computational time of TDC-TAM may be affected considerably. Therefore, to facilitate TDC-TAM integration a test time estimation function is introduced in the following section.

4. Test time estimation

The test time estimation function will provide a correlation between the test set, which has to be compressed, and the external test time. With respect to the XNet compression method, we note $acb =$ the average care bits per WSC load; and $atpl =$ the average test pattern lockouts per WSC load. If acb is known, the external test time is given by:

$$time^{XNet}_{w}(acb) = time_{w} \cdot (1 + atpl_{w}(acb)) \qquad (1)$$

where $time_w$ represents the internal test time for a bus width of w, and $atpl$ is a function of w and acb. In order to estimate the external test time, we have to determine the relation between $atpl$, w and acb. To achieve this we use the following experimental setup: **(i)** we perform controlled static compaction [11], for maximum care bit values per test vector of



(a) $time_{int}$ vs. $time_{ext}$ for s38584 (b) Results for $w = 32$ and $atpl_{32}(x)$ (c) Random test sets: $XNet$ vs. $time_w^{XNet}$

Figure 4. Pareto-optimum points and test time estimation

$B = \{50, 100, 150, 200, 250, 300, 400, 800, 1000, 2000\}$, using the test sets obtained by ATALANTA [1] for one fault per test vector, for all the ISCAS89 benchmark circuits [4]; **(ii)** for each of the cores we perform core wrapper design using the algorithm proposed in [14] for bus widths $w = \overline{8 \ldots 128}$, and map the test set obtained in the previous step accordingly; **(iii)** we use $XNet(w)$ on the mapped test sets and determine the *acb* and $atpl$ for each test set for each test bus; and finally **(iv)** for each w, the average $atpl$ and *acb* among the cores is computed.

The results for $w = 32$ are plotted in Figure 4(b). In order for these results to be useful in a TAM design algorithm where the *acb* may be different than the ones obtained above, we performed curve fitting for each test bus using the function $atpl_w(x) = a_w \cdot x^3 + b_w \cdot x^2 + c_w \cdot x + d_w$. The plot corresponding to $w = 32$ is also illustrated in the figure. To illustrate that (1) provides a good estimate of the external test time, we compute the $time_{ext}$ for the 32 bit bus system in Example 1. Hence, $time_{32}^{XNet} = 15200 \cdot (1 + atpl_{32}(2.64079)) = 41482.47$, which, in comparison to $time_{ext} = 41554$ (see Example 1), is a good estimation. In addition, we implemented a random test set generator, where the care bit density can be specified, and compared the computed ($XNet(w)$) and the estimated ($time_w^{XNet}$) test times. While in general $time_w^{XNet}$ overestimates $XNet(w)$, this behavior is consistent, and hence the two are strongly correlated (see Figure 4(c)). Hence, the estimation function can be used to drive a TAM design heuristic.

To summarize this section, it becomes apparent that reducing $time_w^{XNet}$ is based on minimizing two conflicting parameters. On the one side, one would increase w to reduce to a minimum the number of WSC loads, hence $time_w$, on the other hand, with the increase in w we have an increase in *acb* and hence $time_w^{XNet}$ may actually increase. In the following section we propose an iterative TAM design algorithm which based on the findings in Sections 3 and 4, provides an integrated TDC-TAM design solution.

5. Compression-driven TAM design

Corresponding to the *TAM add-on*–TDC scenario (see Figure 1(c)) we define the TDC–TAM problem for core based SoC test.

TDC–TAM *Given a SoC with N_c cores, and a maximum external test bus of W_{ext} determine an internal TAM design for the SoC such that when used in conjunction with a compression method the external test time is minimized.*

The problem is NP-hard, which can be shown by restricting it to the TAM design problem introduced in [20]. Since the considered TDC method requires only 2 inputs, in the remainder of this work $W_{ext} = 2$. We illustrate next an iterative approach to the TDC-TAM problem, and extend the TAM design algorithm from [20] toward a compression-driven solution.

The main body of the algorithm is illustrated in Algorithm 1. The inputs to the algorithm are the maximum acceptable internal TAM width (W_{int}^{max}), and the cores (C, with $|C| = N_c$). For each core we compute the Pareto-points ($\mathcal{P}_c$) (line 2). For each Pareto-point we generate a test set (T_{c,w_i}) which is then characterized in order to determine acb_{c,w_i} (lines 4 and 5). And, for each $w_{int} = \overline{8 \ldots W_{int}^{max}}$ we apply the compression-driven TAM design algorithm based on the work from [20] (line 9). After each compression-driven TAM design, the compound test set is generated and compressed using $XNet(w_{int})$. The obtained $time_{ext}$ is stored, and the solution with minimum time and w returned.

The work in [20] introduces $k-tuples$ as a mean for large search space exploration. Starting from the $k-tuples$ two constraint graphs are generated: a horizontal constraint graph and a vertical constraint graph; where the nodes represent cores. Two nodes are connected with a directed edge *iff* there is a given order relationship between the cores in the $k-tuples$ representation. The weight of each edge in the graphs corresponds to the test time and test bus width of the starting node, respectively. The longest paths in the two graphs corresponds to the test bus width and test time of the TAM assignment, respectively. The TAM design heuristic in [20] comprises four main steps: **(1)** *initialization* - where the initial TAM assignment is generated; **(2)** *generate next assignment* - where based on the current assignment, a new TAM assignment is generated; **(3)** *validate assignment* - which ensures that the generated assignment meets the w_{int} constraint, by reducing w_i for the cores which will suffer the smallest penalty in test time; and **(4)** *improve assignment* - where for the cores that are not on the longest vertical path,

Algorithm 1 TDC–TAM heuristic

INPUT: W_{int}^{max}, C | **OUTPUT**: $time_{ext}, w$

1. **foreach** $c = \overline{1 \ldots N_c}$ **do**
2. $\quad \mathcal{P}_c = ComputeParetoOptimumPoints(C_c)$
3. $\quad$ **foreach** $w_i \in \mathcal{P}_c$ **do**
4. $\quad\quad T_{c,w_i} = GenerateTestSet(c, w_i)$
5. $\quad\quad acb_{c,w_i} = CharacterizeTestSet(T_{c,w_i})$
6. $\quad$ **done**
7. **done**
8. **foreach** $w_{int} = \overline{8 \ldots W_{int}^{max}}$ **do**
9. $\quad \pi_{w_{int}}$ =PerformCompressionDrivenTAMDesign(w_{int})
10. $\quad \mathcal{T}_{w_{int}}$ =GenerateCompoundTestSet($\pi_{w_{int}}$)
11. $\quad$ **store(**XNet(w_{int})($\mathcal{T}_{w_{int}}$)**)**
12. **done**

w_i is increased. Step **(2)**, **(3)** and **(4)** are executed for a number of iterations, and the best assignment chosen as the TAM design solution.

To extend this approach toward compression-driven TAM design we performed two changes. Firstly, we replaced the edge value in the horizontal constraint graph with $time_{w_{int}}^{XNet}(acb_{c,w_i} * w_{int}/w_i)$, i.e., we compute the external test time per core as if it was the only one driven by the decoder. And secondly, we enhanced step **(3)** and **(4)** as follows. In step **(3)** we reduced the test bus width of the core which has the smallest penalty in test time and the biggest reduction in *acb* – to obtain a valid solution with the smallest *acb*; while in step **(4)** we reduced the width of the cores which are not on the critical horizontal path – to achieve further reduction in *acb* without penalties in test time.

Note that the test time estimation function integrates smoothly with the heuristic from [20]. Hence, it is expected that other TAM design approaches (e.g. [8, 15, 22]), can also be easily extended toward a TDC-TAM solution.

6. Experimental results

In order to confirm the need for compression-driven TAM design, we implemented the heuristics described in Section 5 in C++ and we chose 3 SoCs based on ISCAS89 benchmark circuits [4] as follows: $S_1 = \{s5378, s9234, s13207, s15850, s35932, s38417, s38584\}$, $S_2 = 2xS_1$ and $S_3 = 3xS_2$. The TDC-TAM heuristic has been evaluated for core test sets with different care bit densities, obtained as illustrated in Section 4, on an *AMD Athlon* at 1.2 *Ghz* with 1*Gb* of RAM. In all the cases the TAM design algorithms have run for 10000 iterations.

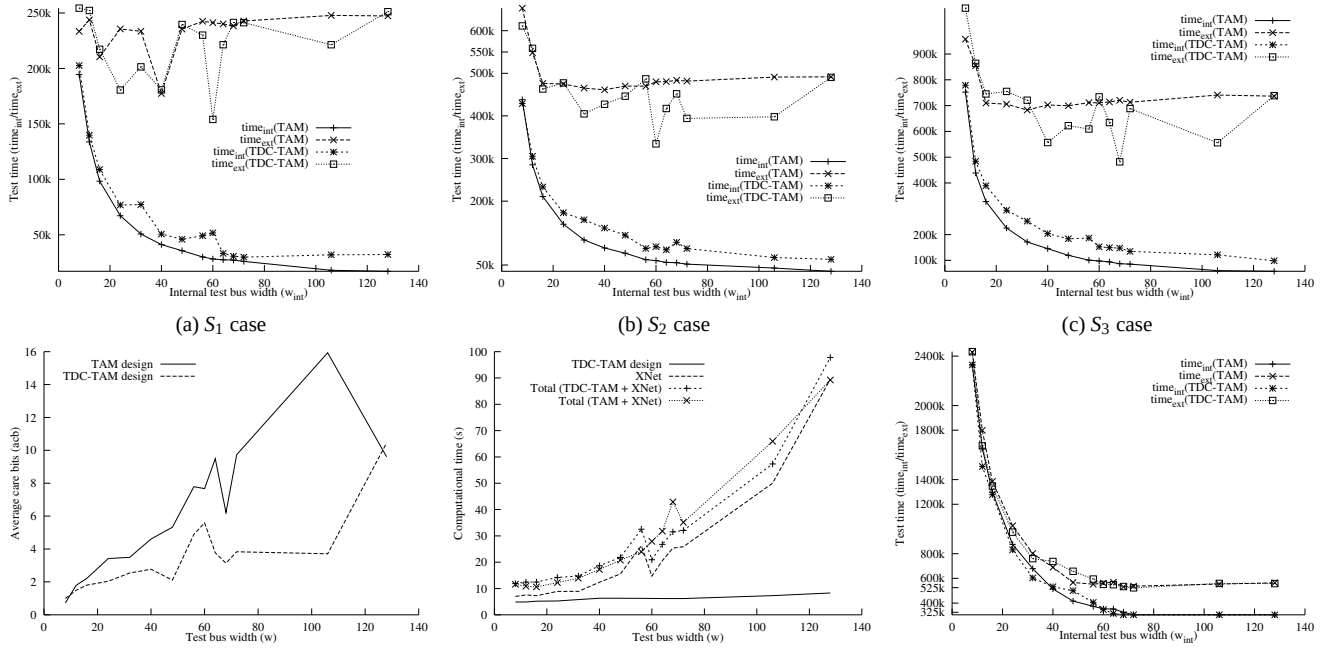
The results are reported in Figure 5. Figure 5(a) illustrates the results for S_1. As expected, the internal test time obtained with TAM design ($time_{int}(TAM)$) steadily decreases with the increase in w_{int}. The corresponding external test time for this case ($time_{ext}(TAM)$) is also shown in the figure. It can be clearly seen that minimum $time_{ext}$ is not obtained for the maximum $w_{int} = 128$ but rather for $w_{int} = 40$. For the TDC-TAM algorithm, we note that the $time_{int}(TDC-TAM)$ is greater than the $time_{int}(TAM)$, however the $time_{ext}(TDC-TAM)$ is generally smaller. For example for $w = 60$, reduction from 250*k* to 150*k* in test time is obtained. Overall, a reduction of $\approx 20k$ in test time is obtained by using the TDC-TAM approach. Figures 5(b) and 5(c) show the results for system S_2 and S_3, respectively. Similar to Figure 5(a), the $time_{int}(TAM)$ steadily decreases with the increase in w_{int}, while $time_{ext}(TAM)$ has significant test time penalties. The performance of the TDC-TAM heuristic appears to be better for these two cases (S_2 and S_3), since external test time ($time_{ext}(TDC-TAM)$) reduction of $\approx 200k$ is obtained in both cases, with a $2x$ reduction in internal test bus width (from 128 to 64). There are two interesting features common to Figures 5(a)–5(c), the $time_{ext}(TDC-TAM)$ plot exhibits a zigzag behavior, and the minimum external test time is usually found in the neighborhood of $w_{int} = 64$. The first feature can be attributed to the random element, part of the TAM design algorithm, i.e., the generation of the next assignment is based on using a random function [20]. The second feature can be attributed to (1) (see Section 4), where the $time_{ext}$ is a function of both $time_{int}$ and *acb*. While, as shown in Figure 5(d), the TDC-TAM design obtains overall smaller care bit density than TAM design, the *acb* does increase with the w_{int}. With $time_{int}$ decreasing and *acb* increasing, for the considered systems, the minimum $time_{ext}$ is obtained for w_{int} in the neighborhood of 64.

Figure 5(e) plots the computational time for the two approaches ($Total(TAM+XNet)$ and $Total(TDC-TAM+XNet)$) in seconds. It can be noted that the differences are very small ($< 10s$), and in general $Total(TAM+XNet) > Total(TDC-TAM+XNet)$. This is due to the fact that the higher the *acb*, the longer it will take to find the correct *SR* values (see Section 3). Also plotted in the figure are the computational times of TDC-TAM and XNet. It is clear from the figure that $XNet$ takes considerably longer than TDC-TAM when *acb* increases.

While the above experiments have been performed for maximum compacted test sets (e.g. $B = 2000$), we also considered cases with very small care bit density ($B = 50$). The results for S_1 and $B = 50$ is given in Figure 5(f). It is important to note that even for small care bit density, there is a significant difference between $time_{int}$ and $time_{ext}$, $\approx 2x$. While the TDC-TAM heuristic does not perform well for these cases, based on these experiments we believe that in core based SoC test, where the system integrator is restrained from performing ATPG, even if the cores are delivered with small care bit density test sets, the system integrator should employ a compression-driven TAM design to efficiently exploit the test bus resources and reduce the test time.

7. Conclusions

When compression methods which are **sensitive** to the test bus width are used on-top of existing TAM design approaches, the result may exhibit inefficient test bus usage and test time penalties. Therefore, in this paper we illustrated the importance of considering the TDC method's



(a) S_1 case (b) S_2 case (c) S_3 case

(d) S_3 average care bits analysis (e) S_2 computational time comparison (f) S_1 case for $(B = 50)$

Figure 5. Experimental results

characteristics in the TAM design and we provided a systematic approach for deriving a test time estimation function based on the test set's care bit density. Finally, we introduced a compression-driven TAM design heuristic to reduce external test time and improve test bus resource usage. Based on experimental validation this paper showed that for test bus **sensitive** compression methods a compression-driven TAM design exhibits better resource usage in comparison to a compression-unaware approach. Therefore, this paper contributes toward test resource optimization when TDC is considered. Future work will analyze the possibility of TDC-TAM integration of cores with multiple compression schemes and fully specified test sets.

References

[1] Virginia Polytechnic Institute and State University. http://www.ee.vt.edu/˜ha/cadtools/cadtools.html.

[2] Bernd Koenemann et al. A SmartBIST Variant with Guaranteed Encoding. In *ATS*, pp 325–330, Nov. 2001.

[3] B. Bottoms. The third millennium's test dilemma. *IEEE Design & Test of Computers*, 15(4):7–11, Oct. 1998.

[4] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *ISCAS*, pp 1929–1934, May 1989.

[5] A. Chandra and K. Chakrabarty. System-on-a-Chip Test Data Compression and Decompression Architectures Based on Golomb Codes. *IEEE TCAD*, 20:113–120, Mar. 2001.

[6] R. Dorsch and H.-J. Wunderlich. Tailoring ATPG for Embedded Testing. In *ITC*, pp 530–537, Oct. 2001.

[7] Erik Jan Marinissen et al. A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores. In *ITC*, pp 284–293, Oct. 1998.

[8] S. K. Goel and E. Marinissen. Layout-Driven SOC Test Architecture Design for Test Time and Wire Length Minimization. In *DATE*, pp 738–743, Mar. 2003.

[9] P. T. Gonciari, B. Al-Hashimi, and N. Nicolici. Integrated Test Data Decompression and Core Wrapper Design for Low-Cost System-on-a-Chip Testing. In *ITC*, pp 64–73, Oct. 2002.

[10] P. T. Gonciari, B. Al-Hashimi, and N. Nicolici. Reducing Synchronization Overhead in Test Data Compression Environments. In *Digest of Papers ETW*, pp 147–152, May 2002.

[11] P. T. Gonciari, B. Al-Hashimi, and N. Nicolici. Test Data Compression: The System Integrator's Perspective. In *DATE*, pp 726–731, Mar. 2003.

[12] P. T. Gonciari, B. Al-Hashimi, and N. Nicolici. Variable-length Input Huffman Coding for System-on-a-Chip Test. *IEEE TCAD*, 22:783–796, June 2003.

[13] ITRS. The International Technology Roadmap for Semiconductors, 2001 Edition. http://public.itrs.net/.

[14] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. Co-Optimization of Test Wrapper and Test Access Architecture for Embedded Cores. In *ITC*, pp 1023–1032, Oct. 2001.

[15] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. On Using Rectangle Packing for SOC Wrapper/TAM Co-Optimization. In *VTS*, pp 253–258, Apr. 2002.

[16] V. Iyengar, A. Chandra, S. Schweizer, and K. Chakrabarty. A Unified Approach for SOC Testing Using Test Data Compression and TAM Optimization. In *DATE*, pp 1188–1189, Mar. 2003.

[17] Janusz Rajski et al. Embedded Deterministic Test for Low Cost Manufacturing Test. In *ITC*, pp 301–310, Oct. 2002.

[18] A. Jas, J. Ghosh-Dastidar, and N. A. Touba. Scan Vector Compression/Decompression Using Statistical Coding. In *VTS*, pp 114–121, Apr. 1999.

[19] A. Khoche and J. Rivoir. I/O Bandwidth Bottleneck for Test: Is it Real ? In *Proceedings of Test Resource Partitioning Workshop*, pp 2.3–1–2.3–6, Nov. 2000.

[20] S. Koranne and V. Iyengar. On the Use of $k-tuples$ for SoC Test Schedule Representation. In *ITC*, pp 539–548, Oct. 2002.

[21] L. Li and K. Chakrabarty. Test Data Compression Using Dictionaries with Fixed-Length Indices. In *VTS*, pp 219–224, Apr. 2003.

[22] E. J. Marinissen and S. K. Goel. Control-aware test architecture design for modular SOC testing. In *Proceedings IEEE European Test Workshop (ETW)*, pp 57–62, May 2003.

[23] E. J. Marinissen, R. Kapur, M. Lousberg, T. McLaurin, M. Ricchetti, and Y. Zorian. On IEEE P1500's Standard for Embedded Core Test. *JETTA*, 18(4):365–383, Aug. 2002.

[24] Y. Zorian and E. J. Marinissen. System Chip Test: How Will It Impact Your Design? In *DAC*, pp 136–142, June 2000.