Compression-based SoC Test Infrastructures

Julien DALMASSO, Marie-Lise FLOTTES, Bruno ROUZEYRE

LIRMM, Univ. Montpellier II/CNRS, 161 rue Ada, 34932 Montpellier cedex 5, France
{dalmasso, flottes, rouzeyre}@lirmm.fr

Abstract.

Test Data Compression techniques have been developed for reducing requirements in terms of Automatic Test Equipments. In this paper, we explore the benefits of using these techniques in the context of core-based SoCs. Test Data Compression is used to reduce the system test time by increasing the test parallelism of several cores without the expense of additional tester channels. In this paper, we first discuss the constraints on test architectures and on the design flow inferred by the use of compressed test data. We propose a method for seeking an optimal architecture in terms of total test application time. The method is independent of the compression scheme used for reduction of core test data. The gain in terms of test application time for the SoC is over 50% compared to a test scheme without compression.

1. Introduction

Testing a SoC mainly consists in testing each core in the system. In order to provide accessibility to these cores, the SoC architecture is completed by a Test Access Mechanism (TAM) and wrappers interfacing cores with the TAM (IEEE 1500 standard [1]). The TAM is generally a bus whose bandwidth fits the number of SoC test IOs. TAM and wrappers are preferably co-designed in order to reduce the global Test Application Time (TAT): several methods formulated as optimization problems have been proposed for establishing the best trade-off between the number of test buses, the bus bandwidth, the wrapper size and the test parallelism (e.g. [2], [3], [4], [5], [6]). However, as the complexity of SoC design keeps on growing, testing becomes more and more expensive with regard to test time and test pin requirements. While increasing the number of scan chains in a core helps to reduce its test time, it also increases the bandwidth of the core interface with the TAM. The consequence of this local test time improvement is to either reduce the test parallelism possibilities at system level, or increase test resources requirements: larger TAM, larger numbers of test inputs and higher requirements in terms of tester channels.

Several Test Data Compression (TDC) techniques aiming at reducing the number of visible scan chains have been developed. Concerning test pattern compression, also called horizontal compression, those techniques consist in compressing test patterns off line (i.e. reducing their bit width), storing the compressed test data in the ATE,

and decompressing test data on-chip for restoring initial test patterns (see Figure 1). Input-data compression schemes rely on the fact that test patterns originally contain don't-care bits. These don't care bits do not have to be stored into ATE but can be supplied on-chip in some other ways. LFSRs [7][8], Xor networks [9] [10], ring generator [11], RAM [12] [13], arithmetic units [14] and test pattern broadcasting among multiple scan chains [15] [16] [19] constitute a range of solutions for minimizing the number of data to be stored into ATE. All these methods reduce therefore the number of necessary ATE channels (W_{ATE}) required to test a standalone core including N scan chains ($N > W_{ATE}$).

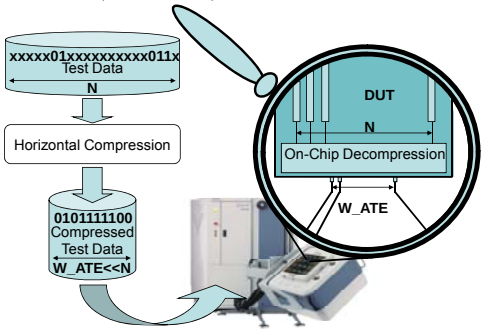


Fig. 1. Compression/Decompression scheme

Note that as mentioned earlier, increasing the number of internal scan chains in a core, and therefore its interface with the TAM, allows reducing its test time since the resulting test scheme requires fewer scan-in clock cycles. However, if a compression technique is used for keeping the number of visible scan chains W_{ATE} as low as possible ($W_{ATE} < N$), the core test time may be affected compared to a solution where the number of visible scan chain is equal to the number of real scan chains ($W_{ATE} = N$). Because no matter the TDC technique is used, compressing an N-bits vector on a W_{ATE}-bits word to be stored in the ATE is not always possible. Consequently, it is necessary to serialize the non-compressible vectors with the help of a decompressor-bypass mechanism, or to look for additional compressible test patterns for keeping the fault coverage obtained with the original non-compressed test sequence. In any case, a side-effect is an increase in TAT of the core under test.

Concerning test responses, several methods have been proposed (e.g. [17], [18]). Conversely to TDC, those test responses compaction techniques do not impact TAT and are independent of the core netlist and of the test responses sequence. Thus, they can be directly employed in the framework of SoCs design. In the remainder of this paper, we focus on test pattern compression only.

Several TDC approaches can be considered at system level. In a bottom-up approach, TDC is applied at core level by the core designer, and then wrapped cores (including decompressors) are embedded in the SoC by the system integrator. In this case, the test infrastructure design resumes to the classical TAM optimization problem since individual test times and number of visible scan chains on the core interfaces are known, and fixed, before system integration. The second approach consists in questioning test time optimization and compression schemes at system level. In this case, the cores come with their uncompressed test sequences and the

system integrator must determine the compression ratio on every core, define the test infrastructure and resulting test time. This approach should allow optimizing the test of the system with regard to the test resources constraints and not only with regards to the pre-fixed test times of the individual cores as in the bottom-up approach.

Concerning TDC at system level, several approaches focus on memory depth requirements using different forms of stream compression (e.g. [19], [20]), or on test pattern broadcasting among multiple cores (test time savings up to 23% are reported in [21]). TAM architectures using horizontal compression have been presented in [23], [24], [25] but the proposed methods rely on specific TDC techniques. Moreover, all architectural solutions are not considered since these techniques essentially target TAM architectures with a single decompressor for all cores, or architectures with a dedicated decompressor per core (or connected to duplicated versions of the same core).

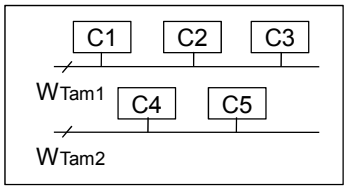
In this paper, we propose a method for exploring all TAM/TDC architectures including solutions with a dedicated decompressor per core and architectural solutions with shared decompressors. The final goal is to generate test architectures and test schedules that minimize the system TAT. The proposed technique is independent of the adopted compression scheme.

Section 2 discusses the implication of TDC insertion at SoC level. The problem formulation as well as notations are given is Section 3. The algorithm is detailed is Section 4 whereas experimental results are reported in Section 5. Finally, Section 6 draws some conclusions.

2. SoC test architecture and compression

2.1 Test infrastructure design

A SoC test architecture is proposed by the IEEE 1500 Standard. It mainly consists of a TAM bus and wrappers around cores. The TAM links the SoC's test IOs to the cores. Each core wrapper interfaces the core and the TAM bus. As in [2] and [3], we assume a TAM architecture organized around a partitioned test bus, each core being connected to one sub-bus, as depicted in Figure 2 in which the TAM is split into two sub-buses TAM1 and TAM2. Cores connected to the same sub-bus are tested serially (e.g. C1, C2, C3), cores assigned to different TAMs can be tested in parallel (e.g. C1 and C4 or C1 and C5). We do not make any assumption about the wrappers of the cores : they can be designed when building the test infrastructure at system level or pre-defined by the 1500-ready cores. In the rest of this paper, W_{TAM} denotes the TAM bandwidth, and W_{TAMi} the bandwidth of sub-bus i.



$W_{ATE}=W_{Tam}= \sum W_{Tami}$

Fig. 2. TAM architecture

Let's recall that, under the chosen TAM model, building the test infrastructure mainly consists in: 1) finding a partition of the bus into p sub-buses and determining their bandwidth, 2) assigning the cores to the p sub-buses, and designing their wrappers 3) deriving a test schedule so that the total test time is minimized. An underlying data of these tasks is the test times of cores.

The test time of a core depends, among other things, on the size of its wrapper in terms I/Os interfaces with the TAM. The test time of the core and its wrapper size are linked by the following relation:

$$T_{core} = V \times [1+\max\{si,so\}] + \min\{si,so\}$$

where V is the number of test patterns, and si (so) the number of scan cycles required to load (unload) a test vector (test response). In Figure 3 for instance, two wrappers configurations of the same core are depicted. On the left hand side, the core is connected to the TAM through 2 visible scan chains and the test time is 13p+12 cycles. On the right hand side, the wrapper interface is enlarged to 3 scan chains at the benefit of the test time, which is reduced to 10p+6 cycles.

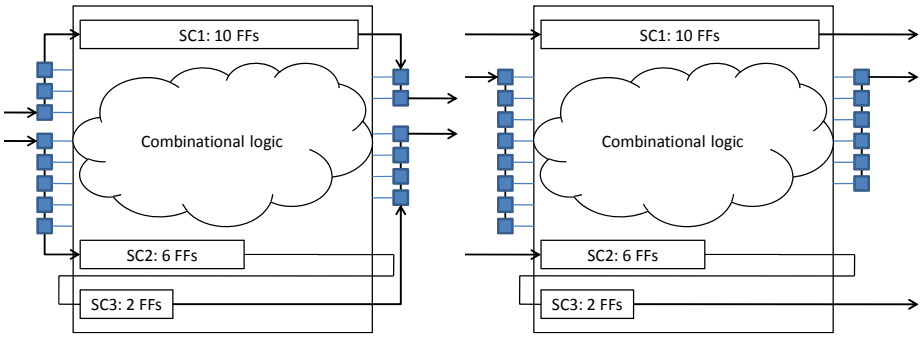


Fig. 3. Wrapper designs

2.2 Compression and test infrastructure

The TAM bandwidth can be increased thanks to TDC techniques without changing the requirements in terms of ATE channels (W_{ATE}). The tests parallelism can therefore be increased without additional cost and should result in a shorter test time. However, as explained in the introduction, TDC may also increase the test times of individual cores. More precisely, for a fixed number N of scan chains (or equivalently

for a fixed wrapper size), the test time of a core increases when the number of bits at the input of the decompressor gets smaller. For instance, using the TDC technique presented in [14], the test of the S38417 benchmarks circuit with N=16 scan chains needs 21451 clock cycles when $W_{ATE} = 10$ and increase to 38867 clocks cycles when $W_{ATE} = 3$ (see for instance, results given in Figure 9).

The use of TDC impacts the building of the test infrastructure in two aspects:

- 1) Since TDC modifies the test times of individual cores, the decompression ratios must be established *during* the design of the test infrastructure and not after.
- 2) Since decompressors can be shared between several cores, test sequences to compress must be defined before decompressor assignation.

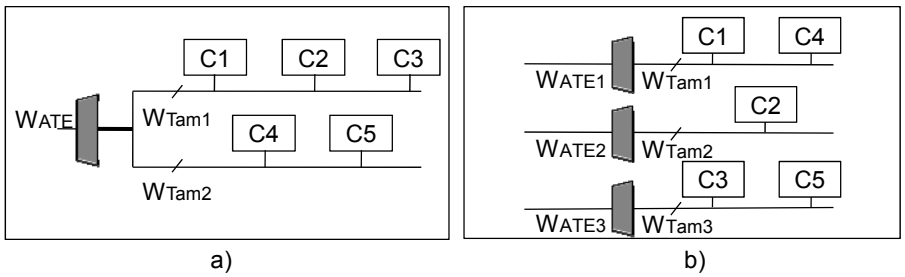


Fig. 4. TAM/decompressors architectures

Let's discuss this last point on the Figure 4 example: either a decompressor feeds several sub-buses (Figure 4.a) or one decompressor feeds a single sub-bus (in Figure 4.b). The evaluation in terms of test time of a solution requires defining the bus partitioning, the core assignment, the test parallelism, the test sequence, and finally the compression of this sequence. Figure 4.a for instance depicts only one bus partitioning and core assignment possibility. It includes several test parallelism solutions (e.g. either C1 and C4 tested in parallel or C1 and C5). In turn, each one necessitates building the actual test sequence by concatenating the test sequences of the cores tested in parallel, C1 and C4 for instance. Finally the resulting test sequence has to be compressed in order to obtain the actual test time. Another way of dealing with this model is 1) to build the optimal test infrastructure and related test schedule without looking at compression 2) derive the whole SoC test sequence and compress it. Doing so, there is no chance to obtain an optimal solution since the test infrastructure (without decompressor) is built given the original test times of cores which are latter modified by the compression. We did such an experiment with the example given in section 5. Doing so, the obtained test time is 65699 cycles while a solution with 57941 cycles has been obtained using the method we propose here.

Conversely, the cores connected downstream a decompressor in the second architecture style (Figure 4.b) are tested one after the other. The test sequences to compress are simply those of the cores and not issued from the concatenation of several ones. The compression of the test sequences can therefore be done independently of the test infrastructure building process. This alleviates the problems raised by the first model.

So in the remaining, we consider the second architecture style and we propose a method for *conjunctly* building up the TAM, the wrappers (if needed) and the decompressors.

Each path for the ATE channels, through a decompressor, up to sub-bus is called a *line*. The architecture depicted in Figure 4.b) is composed of three lines for instance. It must be noted that within this architectural model and in the absence of additional constraints such as power limit for instance, the test scheduling is trivial (as without compression). The test time on a line is simply the sum of the individual test times of the cores since there is no test parallelism on the line. The total TAT at system level is the maximal test times over the lines.

3. Problem statement and notations

We state the problem of building the test infrastructure with decompressors as an optimization problem. Given the number of available ATE channels, the bandwidth of the TAM, and the test patterns, we want to determine the best partition of the test infrastructure into p lines and the interconnection of the cores to the sub-buses so that the TAT is minimized.

In the remaining, we will use the following notations. the ratio W_{ATE}/N is denoted by ρ. n is the number of cores under test, w_c the number of visible scan chains for every core c=1…n. Let p be the number of lines, and let W_{ATE_i} and W_{TAM_i}, i=1,…,p be respectively the number of ATE channels and the bandwidth of the sub-bus on line i. Let $\rho_i=W_{ATE\_i}/W_{TAM\_i}$ be the decompression ratio of line i. $t^c_{w_c,\rho}$ denotes the test time of core c with a w_c bits wrapper for a ratio ρ.

The problem is to determine:

- the line number p;
- the bitwidths W_{ATE_i} and W_{TAM_i} for i=1,…,p;
- an assignment of the cores to the lines;
- optionally, the wrapper size w_c of each core,
- and a test schedule so that TAT is minimal.

The following constraints must be obeyed:

$$W_{ATE} = \sum_{i=1,..,p} W_{ATE\_i} \qquad \text{cons.1}$$

$$W_{TAM} \geq \sum_{i=1,..,p} W_{TAM\_i} \qquad \text{cons.2}$$

$$W_{TAM\_i} \geq W_{ATE\_i}, \text{ i =1,..,p} \qquad \text{cons.3}$$

$$w_c \leq W_{TAM\_i} \text{ if c is connected to line i.} \qquad \text{cons.4}$$

Variables to be determined are given in italic in Figure 5 (.W_{ATE} and W_{TAM} being given)

Concerning core wrappers, there are two cases: either the cores are wrapper-ready or their wrappers have to be designed. In the later case, it must be noted first that $1 \leq w_c \leq \max(\#PIs,\#POs) + \#\text{scan chains}$. Secondly, once w_c is determined, designing the wrapper so that the test time of the core is minimized resumes simply to balance the lengths of the visible scan chains. This won't be detailed in the remaining.

In the scenario where the wrappers are already fixed, if a core is assigned to a line i for which W_{TAM_i} is strictly greater than the wrapper size w_c, only w_c bits of TAM_i

are connected to the wrapper, the test time of the core is considered to be the same as if TAM_i was w_c bits wide. For instance, and for a core c with w_c =4, its test time $t^c_{w_c,\rho}$ is the same whether it is assigned to a line with W_{ATE} = 2 and W_{TAM} =6, or to a line with W_{ATE} = 2 and W_{TAM} =4, i.e. ρ = 2/4.

The test time $t^c_{w_c,\rho}$ of a core c must be pre-computed for all possible values of ρ (from 1 to $1/w_c$). (cf. section 4 to see how this process can be speeded up). For examining the benefit of using TDC when designing the test infrastructure of a SoC, we developed the heuristic presented hereafter.

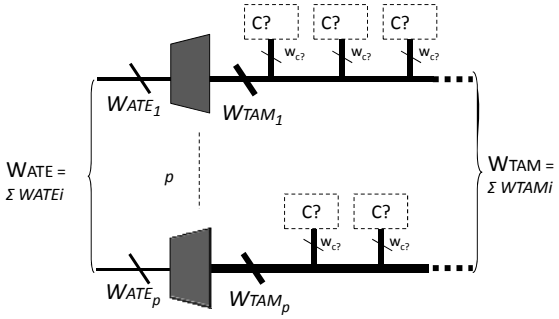


Fig. 5. Problem statement

4. Algorithm

The general flow chart of the method is depicted in Figure 6. First, all the possible combinations of lines are explored (line 1 and 2). The ATE channels partition can be easily determined knowing the total W_{ATE} width and the number of lines p by applying the formula of the partition of integer numbers. Namely, the number X(n,p) of partitions of a set of n elements into p subsets can be computed as:

$$X(n,p) = \sum_{k=1}^{p} X(n-p,k) \text{ with } X(n,n) = X(n,1) = 1 \quad \textbf{(1)}$$
$$\text{and } X(n,p) = 0 \quad \text{if } p > n$$

For instance, 10 ATE channels can be partitioned into p=3 subsets in X(10,3)=8 different ways (1+1+8, 1+2+7, 1+3+6, etc…).

Then for each ATE channels partition, all the compatible partitions of the TAM are calculated. A partition of the TAM is said to be compatible with a partition of the ATE channels if cons.3 is verified for all p lines. Furthermore, if cores are wrapper-ready i.e. w_c are fixed, the number of TAM partitions to be explored can be further reduced by considering cons.4. In other words, the narrowest TAM must be large enough to support the narrowest wrapper. It must be noticed that if W_{TAM_i}=W_{ATE_i},

no decompressor is present on this line. For a pair of partition, (ATE channels partition and TAM partition), cores must be assigned and the scheduling performed to obtain the TAT of this architecture (line 3 in Figure 6).

1. For all ATE channels partitions into p parts
2. For each compatible TAM partition into p parts
3. Find the best assignment of the cores to the p lines (that minimize TAT) ->cf Fig 5.

If this assignment reduces the global TAT, memorize this assignment and ATE/TAM architecture

Fig. 6. Partition algorithm

Seeking for the assignment of cores to lines that minimizes TAT is an NP-complete problem. So we developed the heuristic given in Figure 7.

// Initial Solution
- Sort cores by decreasing test data volume
- Assign each core to the largest bus so that TAT increases as few as possible.

// Improvement of the solution
- While TAT is reduced
  - Find the line i with the highest TAT_i
  - For each core c assigned to i,
    - For all other lines k ($k \neq i$)
      - Move core c from i to k
      - Compute newTAT and memorize i, k, c and newTAT
      - Move back core c from k to i
  - Move core c from i to k such that:
    1) the smallest TAT has been obtained
    2) the number of useless bits on k is minimized
    3) the standard deviation between TAT_i of all lines is maximized

Fig. 7. Assignment algorithm

The first step determines an initial solution of the architecture, i.e an initial assignment of cores to the TAMs. Each core is positioned on the largest possible TAM i.e. and its wrapper size is set according to (cons.4). If the core is wrapper-ready, it is assigned to the smallest bus i.e. respecting cons.4. For instance, in case of 3 TAMs having resp. 5, 7 and 10 bits, a core with a 6-bits wrapper will be assigned to the 7 bits TAM. The first bus is not large enough to be connected to the core's

wrapper (cons.4). The second bus is preferred to the third one since, a priori, it is beneficial to reserve the larger one for cores with larger wrappers.

The second step consists in improving this initial solution. For this, the cores are moved to other lines to reduce the global TAT.

The principle is to move a core from the line with the highest TAT to another line so that the global TAT gets reduced as much as possible. For that, all cores of the line are virtually shifted to other lines and TATs are computed accordingly. The move that gives the highest benefit is chosen. In case of equality, the algorithm chooses (Core c, Line i) such that the number of useless bits on the line is minimized i.e. $W_{TAM\_i} - w_c$ is minimal. This is done for getting more room to move cores with larger wrappers to large buses, in next steps. Similarly, a third order criterion is used to unbalance test times over lines.

Let's recall that the computation of TAT is straightforward (TAT_i denotes the test application time on line i):

$$TAT = \max(TAT_i, i = 1,..., p) \text{ and } TAT_i = \sum_{\text{cores assigned to i}} t^c_{w_c,\rho_i} \quad (2)$$

Note that the test times $t^c_{w_c,\rho}$ for all cores and for all compression ratios (W_{ATE}_i/w_c) are inputs of the proposed algorithm. These data are necessary to compute the system TAT (i.e. schedule the tests). Thus, as a pre-process, the compression algorithm must be performed for all compression ratios, for all cores and all wrapper sizes. This can be very CPU expensive depending on the compression technique used. We propose here an alternative to the exhaustive computation

First, when the wrapper size is questioned, let's recall that as reported by many authors, the test time of a core, in the absence of compression i.e. $\rho=1$, is a stepwise decreasing function of the wrapper size. Furthermore it depends on the number of test vectors and not on the vectors themselves. Figure 8 reports the test time versus w_c for the 10th core of the D695 ITC'02 benchmark. In general the number of steps is small. Only 15 optimal values of w_c have to be considered for this core.

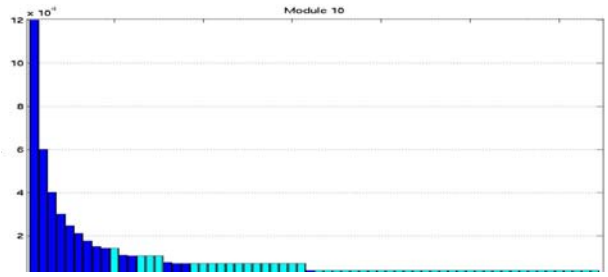


Fig. 8. ITC'02 d695 benchmark (core 10) Test time vs wrapper size

Secondly, whatever the TDC technique is used, the same behavior of the test time of cores versus decompression ratio can be observed (for a given wrapper size w_c). It can be identified to the function:

$$t^{c}_{w_c,\rho} = \frac{\alpha}{\rho} + \beta \quad (3)$$

Only two values of t for one core are sufficient to identify α and β. The estimated values of $t^{c}_{w_c,\rho}$ for several decompression ratios are thus obtained from only two measured values instead of w_c computations. In order to improve the precision of the estimation, the compression algorithm is performed with the first and last decompression ratio values.

This property has been validated with the TDC method [14]. This compression scheme is applicable with intellectual property cores and it is Test Suite independent, i.e. it does not required specific test generation or fault simulation.

The measured and estimated $t_{c,\rho}$ values are reported on Figure 9 for the ISCAS'89 s38417 benchmark (16-bits wrapper). The maximum error between measured and estimated values is smaller than 1%. Similar results have been obtained for all ISCAS'89 benchmarks and several configurations of wrappers.

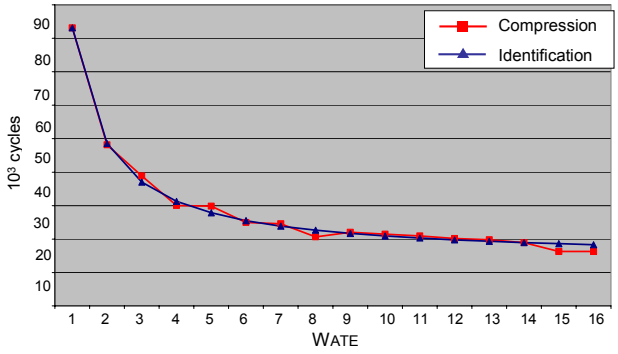


Fig. 9. S38417 (w_c = 16) computed/estimated test times

As a final remark let's note that the proposed heuristic can be adapted to additional constraints such as power limit, precedence constraints, etc…. Concerning the power consumption constraint for instance, two levels of optimization can be envisaged. At core level, the don't care values not assigned by the compression scenari can be assigned in such a way that power consumption is limited during scan shifting. At system level, core test parallelism is not totally fixed by our architectural solution since cores assigned to the same line must be serially tested but there is no constraint on the test order. In Figure 2 for instance, cores on the first line can be tested in the following order C1, C2 and C3 or C1,C3 and C2 for instance. The best solution in terms of power consumption depends of the test order on the second line C4, C5 or C5, C4.

5. Results

The first SoC used for experiments is the one described in [9][23] and depicted in Figure 10. It is composed of 16 ISCAS'89 benchmark circuits used as cores (i.e. with wrappers).

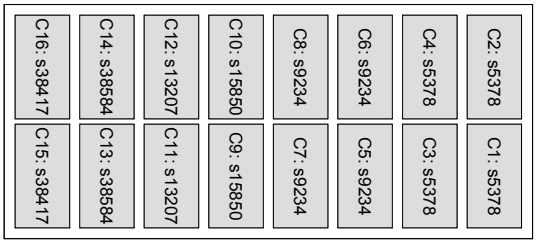


Fig. 9. SoC example from [23]

The test sequences of the circuits have been obtained with the Synopsis ATPG tool TETRAMAX [26] and compressed with our TDC technique described in [14]. The characteristics of the cores are given in table 1.

Table 1. Characteristics of the cores

Core number	#scan chains (wc)	test cycles
1 to 4	5	9331
5 to 8	6	9030
9, 10	10	8804
11,12	12	16048
13,14	14	19845
15,16	16	45760

As explained before, the proposed method able to deal with either wrapper ready cores or with cores for which the 1500 wrapper has to be designed. The 2 following sub-sections present experimental results in both cases.

5.1 Fixed wrapper

In a first series of experiments, we assume that the wrappers are already designed. Wrappers sizes are equal to the number of scan chains. We have set the number of ATE channels to 32 and the maximal total TAM bitwidth to 64. The algorithm has been applied with a number of lines ranging from 2 to 6. Results are reported in Table 2.

Table 2. Architectures exploration results

#lines	# conf.	TAT	Lines' parameters (W_{ATE_i} / W_{TAM_i})	#bits used on TAM
2	522	127413	(16,16) / (16, 48)	30
3	44639	90457	(8,9,15) / (14,16,34)	42
4	1345142	68361	(5,7,8,12) / (7,14,16,27)	53
5	18605924	57941	(5,5,7,7,8) / (6,12,14,16,16)	64
6	142238520	57941	(1,4,5,7,7,8) / (1,5,12,14,16,16)	63

Col.2 indicates the number of architectural configurations that have been explored while col.3 gives the TAT of the elected architecture. The details of the test infrastructure are given in col.4. The last column indicates the actual number of TAM bits.

For instance, for architecture with 3 lines, 44639 configurations have been explored. The optimal one leads to a TAT equal to 90457 test cycles. The architecture is composed of 3 lines with 3 decompressors such that (W_{ATE_1}, W_{TAM_1}) = (8,14), (W_{ATE_2}, W_{TAM_2}) = (9,16), and (W_{ATE_3}, W_{TAM_3}) = (15,34).

From this table, some observations can be done:

- All potential test infrastructures are explored including those that do not contain decompressors. For instance, for the 2 lines configuration, the optimal architecture does not include a decompressor on the first bus $W_{ATE\_1} = W_{TAM\_1}$=16.

- While a budget of a 64 bits TAM has been given, all those bits are not necessarily connected to cores (and thus are useless). This is the case for p=2, 3, 4, 6. This is mainly due to the wrapper sizes chosen for the cores. This means that the actual bitwidth of the TAM is smaller than 64 bits.

Among all compressor/TAM architectures, the best TAT is obtained with p=5 lines. The corresponding test schedule and architecture are given in Figure 11 and Figure 12. Test parallelism cannot be fully exploited with smaller values of p since at most p cores can be tested in parallel. For larger values of p (6, 7, …), further experiments have shown that TAT increases.

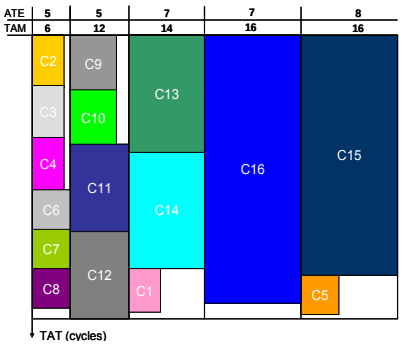


Fig 11 Test schedule for a 32 → 64 bits decompression with 5 lines
TAT = 57941 cycles

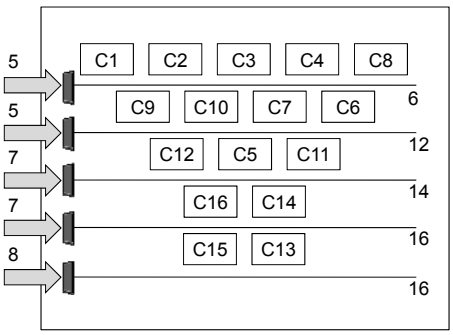


Fig. 12 Final architecture

The reason is that the sizes of the wrappers relatively to the possible TAM_i widths act as a brake on parallelisation.

We measured the benefit of using compressors in SoCs test architectures by comparing them to standard architectures i.e. without using compression, while setting the same environmental constraints. In the first case, we assumed the same limit on the numbers of available ATE channels (32 bits and thus a TAM of 32 bits), in a second case, the same area budget for building the TAM (64 bits wide and thus 64 ATE channels).

For the first case, the TAT is 127413 cycles for a standard TAM architecture when a number of sub-buses p ranges from 2 to 4 and 131210 cycles when p equals 5 or 6. These results have to be compared with the 57941 cycles when compression is used. Thus, the use of TDC technique in the context of SoC infrastructure design leads to a gain of 54.5% in terms of TAT for this example (at the expense of area overhead: larger TAM, decompressors).

In the second case, i.e. a TAM of 64 bits (which means 64 ATE channels for a standard architecture vs 32 ATE channels with compression), comparative results are reported in Table 3. At the evidence, TDC has allowed to divide by two the number of ATE channels at the expense of only a 4% increase on TAT.

Table 3. Architectures Comparison (fixed wrappers)

	Proposed architecture: W_{ATE} =32, W_{TAM}=64		Standard architecture: W_{ATE} = 64, W_{TAM}=64	
# lines	TAT	actual TAM bitwidth	# lines	TAT
2	127413	30	2	127413
3	90457	42	3	90457
4	68361	53	4	68361
5	57941	64	5	57941
6	57941	63	6	55738

5.2 Unfixed wrapper

We did the same experiments, but without assuming fixed wrappers size i.e. letting the method determines the most adequate wrappers structures. TAT are reported in Table 4. It can be fist noted that since wrappers structures are questioned, bus width can be better utilized leading to shorter TAT. Secondly, as in the previous case, the use of TDC leads to a large TAT improvement.

The same kind of experiments has been performed on the g1023 ITC'02 benchmark. Unfortunately, in the ITC'02 suite, neither cores netlists nor test patterns are provided, all information necessary to perform compression. Only the number of test vectors is specified. We have randomly chosen test sequences including the given number of vectors. To be conservative, the patterns are such that they include 80% of don't care bits (many authors report a don't care bits percentage ranging from 95% to 99% on industrial circuits). Comparative results are given in table 5.

Table 4. Architectures Comparison

p	32→64 Decomp. Architecture	Standard 64 bits Architecture	Standard 32 bits Architecture
2	66596	52953	97216
3	61277	49814	96624
4	57337	49129	96736
5	55101	48592	96624
6	54140	48517	96563

Table 5. g1023 Comparison results

	Decomp. Architecture		Standard Architecture		
p	32→64	16→32	64 bits	32 bits	16 bits
2	17492	26256	15153	19633	33952
3	14185	23084	11274	17892	33718
4	12996	21409	11274	17235	33824
5	12399	20719	11274	17215	33824
6	12138	20667	11274	17235	33824

6. Conclusion

In this paper, we explored the benefits of horizontal test data compression techniques in the context of the design of SoC test infrastructures. The increase in parallelism allowed by compression is fully exploited to reduce the test application time of the SoC. We propose a method that explores all architectural solutions from one single decompressor for all cores to architectures with a dedicated decompressor per core. Results obtained on a SoC based on ISCAS'89 benchmarks circuits have confirmed this TAT reduction with a ratio of more than 50%. While the experiments have been performed using a particular TDC technique, the method is independent of the used TDC.

Presently, this method is geared to minimize the test time. Area overhead induced by decompressors and TAM is not taken into account. Seeking the best trade-off is a direction for future research.

References

[1] IEEE standard for embedded core test – IEEE Std. 1500-2004.

[2] V. Iyengar et al.. "Test wrapper and test access mechanism co-optimization for

system-on-a-chip". J. Electronic Testing, vol. 18, no. 2, pp. 213-230, April 2002
[3] V. Iyengar et al., "Efficient Wrapper/TAM Co-Optimization for Large SOCs", DATE'02, pp: 491-497.
[4] V. Iyengar et al. "Wrapper/TAM co-optimization, constraint-driven test scheduling, and tester data volume reduction for SOCs", DAC '02. pp: 685-690.
[5] S.K. Goel, E.J. Marinissen, "Effective and Efficient Test Architecture Design for SOCs", ITC'02, p: 529- 535.
[6] G. Zeng, H. Ito, "Concurrent core test for SOC using shared test set and scan chain disable", DATE'06, pp: 1045-1050.
[7] A. Jas, B. Pouya, N.A. Touba, "Virtual Scan Chains: a means for reducing scan length in cores", VTS'00, pp: 73-78.
[8] L-T Wang et al., "VirtualScan: a new compressed scan technology for test cost reduction", ITC'04, pp: 916-924.
[9] I. Bayraktaroglu, A. Orailoglu, "Test volume application time reduction through scan chain concealment", DAC'01, pp: 151-155.
[10] K.J. Balakrishman, N.A. Touba, "Reconfigurable linear decompressor using symbolic Gaussian elimination", DATE'05, pp: 1130-1135.
[11] J. Rajski et al., "Embedded deterministic test for low cost manufacturing Test", ITC'02, pp: 916-922.
[12] L. Li, K. Chakrabarty, N. A. Touba "Test data compression using dictionaries with selective entries and fixed-length indices", ACM TODAES, Vol. 8, No. 4, October 2003, pp: 470-490.
[13] A. Würtenberger, C.S.Tautermann, S.Hellebrand, "Data compression for multiple scan chains using dictionaries with corrections", ITC'04, pp: 926-935.
[14] J. Dalmasso, M.L. Flottes, B. Rouzeyre, "Fitting ATE Channels with Scan Chains: a Comparison between a Test Data Compression Technique and Serial Loading of Scan Chains", DELTA'06, pp: 295-300.
[15] N. Sitchinava et al., "Changing the scan enable during shift", Proc. VTS'04, pp: 73-78.
[16] H. Tang, S.M. Reddy, I. Pomeranz, "On reducing test data volume and test application time for multiple scan chain designs", Proc. ITC'03, pp: 1079-1088.
[17] S. Mitra, K.S. Kim, "X-compact, an efficient response compaction technique for test cost reduction", ITC 02, pp: 311-320
[18] J. Rajski,et al., "Finite memory test response compactors for embedded test applications", IEEE Trans. on CAD, April 2005, Vol. 24-4, pp: 622- 634.
[19] A. Chandra and K. Chakrabarty, "Test Data Compression and Test Resource Partitioning for System-on-a-Chip Using Frequency-Directed Run-Length (FDR) Codes," IEEE Trans. Computers, vol. 52, no. 8, Aug. 2003, pp. 1076-1088.
[20] P.T. Gonciari, B.M. Al-Hashimi, and N. Nicolici, "Variable-Length Input Huffman Coding for System-on-a-Chip Test," IEEE Trans. Computer-Aided Design, vol.22, no. 6, June 2003, pp. 783-796.
[21] A. Larsson, E. Larsson, P. Eles, Z. Peng, "SOC Test Scheduling with Test Set Sharing and Broadcasting", Proc. ATS'05, Session A4: SoC Testing
[22] B. Arslan, A. Orailoglu, "CircularScan: a scan architecture for test cost reduction", DATE'04, pp: 1290-1295.
[23] V. Iyengar, A. Chandra, "A Unified SOC Test Approach Based on Test Data Compression and TAM Design", Proc. IEEE DFT'03, pp: 511-518
[24] P.T. Gonciari, B.M. Al-Hashimi, "A Compression-Driven Test Access Mechanism Design Approach", ETS'04, pp: 100-105.
[25] P.T. Gonciari, B.M. Al-Hashimi, N. Nicolici, "Integrated Test Data Decompression and Core Wrapper Design for Low-Cost System-on-a-Chip Testing", ITC'02, p: 64-70.
[26] www.synopsys.com/products/test/tetramax_ds.html