Systems-on-Chip: Use of Test Data Compression Technique to Reduce Test Time

Julien DALMASSO,
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
61 rue Ada, 34932 Montpellier CEDEX 5, France
dalmasso@lirmm.fr

Abstract— During the production phase of microelectronics systems, one of the fundamental step is to check if the system works fine. This step is called the test of integrated circuits. Furthermore, cost reduction of these tests has become a major axe of research in microelectronics. Several techniques allow to reduce these costs, whether by reducing the test data volume (vertical compression) whether by reducing the need in Automatic Test Equipment to send the test data to the circuit under test (horizontal compression). This is called Test Data Compression. But most of these techniques can only be applied to single cores. Yet actual Systems-on-Chip are now composed of numerous cores. This paper first presents a new horizontal compression method that is perfectly applicable in the multiple-cores systems framework. Then, two applications of this method to systems (1/bus-based and 2/Network-on-Chip-based systems) are presented. Compression allows here to increase test parallelism and so to decrease Test Application Time of the whole system, and thus test costs.

I. INTRODUCTION

During design and manufacturing phases of integrated circuits, it is essential to check if a circuit works fine or not. As the complexity of Systems-on-Chip (SoCs) designs keeps on growing, testing becomes more and more expensive with regards to Test Application Time (TAT) and test pin requirement. In this context, a common technique for reducing test time is to use multiple scan chains. Indeed, test time of a circuit is inversely proportional to the number of scan chains. Nevertheless, this possibility is limited by the number of available Automatic Test Equipment (ATE) channels. These Test Data Compression (TDC) methods (often referenced as horizontal compression) aim at reducing the number of required ATE channels to feed the scan chains. Some other methods (vertical compression) reduce the amount of test data per ATE channel but are not under the scope of this paper. A basic idea to fit the number of scan chains with the number of ATE channels is to serialize test data (Figure 1-a). This technique requires almost no area overhead but is very expensive in terms of test time since each test slice will be sent serially in several words. Another solution to fit the ATE channels with the scan chains is to use an on-chip decompressor between the ATE and the CUT that will recreate the original data from the compressed one (stored in the ATE memory).

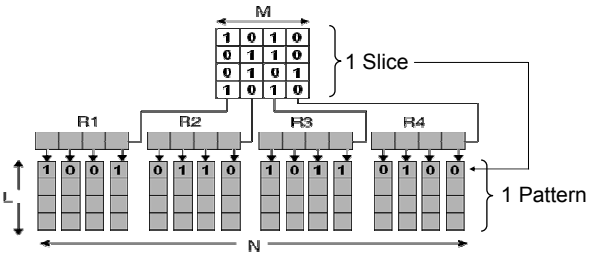

Figure 1-a: Serial loading of test data

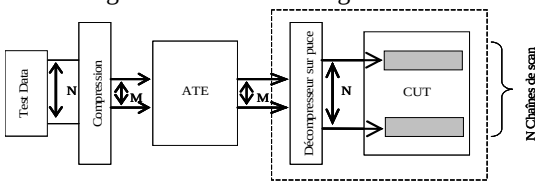

Figure 1-b: General compression-decompression scheme

In the following, we call a *slice* the test values of a given test pattern (or test vector) loaded into the flip-flop of same ranking of the scan chains. Each test pattern is composed of L slices, L being the maximum number of flip-flops in the scan chains.

Several horizontal decompression methods have been proposed but they present one or more of the following drawbacks:

- the scan chains structure depends on the compression technique
- the decompressor structure depends on the test vectors
- they require specific test sequence
- they are based on fault simulation or on ATPG which makes them not suitable for IPs

In section 2, we present a technique that gets rid of these drawbacks. Then in section 3, two applications to Systems-On-Chip are presented.

II. TEST DATA COMPRESSION

A. Principle

The general principle of an horizontal compression-decompression architecture for a circuit with N scan chains, is to send the N bits slices with only M ATE channels to an on chip decompressor that recreates exactly the original slices on N bits. The original test data are compressed and

stored in the ATE memory. During test, the ATE sends the compressed slices to a small M to N on chip decompressor that feed the N scan chains.

The architecture of the decompressor is depicted on Figure 2. It is basically made of an adder and of an accumulator/shifter. The principle is to send the arithmetic difference between two slices (coded on M bits) instead of the N-bits slices themselves.

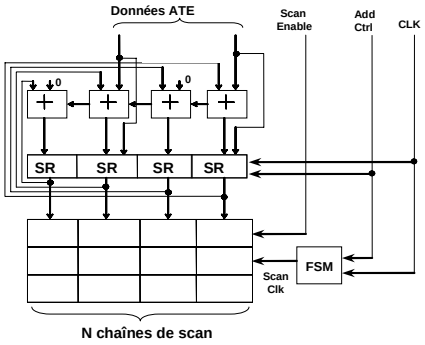

Figure 2: Decompressor architecture, N=4 et M=2

Let S_i be a slice of the test sequence and S_{i+1} the following. The compressed slice is noted Sc_i. It is sent instead of S_{i+1} and is defined by: $S_{i+1} = S_i + Sc_i$.

In the example given in Figure 3, the test sequence T is compressed in T' containing the Sc_i (for N=8 and M=2). T* denotes the test sequence that will actually be applied to the circuit after decompression. Note that the first slice of T' is not compressed but serially loaded in the register, as explained in next section. Of course, depending on the values in S_i and S_{i+1}, it may happen that no compressed slice Sc_i exists. In this case, the register is initialized with the slice S_{i+1}. As with serial loading, the slice S_{i+1} is stored in the ATE as $\lceil N/M \rceil$ words. These M bits slices are sent to the scan chains by a serial loading, bypassing the adder of the decompressor as depicted in Figure 2.

T:			T':		
S_1:	x x 0 0	x 0 1 1	Sc_1:	1 1 0 0	1 0 1 1
S_2:	x 1 0 x	1 x x 0	Sc_2:	1	1
S_3:	1 x x 0	x x x 1	Sc_3:	1	1
S_4:	x x 1 1	x 1 x x	Sc_4:	1	0
S_5:	1 1 x x	x x 1 0	Sc_5:	0	1
S_6:	x 0 x 0	1 x 1 x	Sc_6:	1	0

T*:	
S_1:	1 1 0 0 1 0 1 1
S_2:	1 1 0 1 1 1 0 0
S_3:	1 1 1 0 1 1 0 1
S_4:	1 1 1 1 1 1 0 1
S_5:	1 1 1 1 1 1 1 0
S_6:	0 0 0 0 1 1 1 0

Figure 3: example of a compressed test sequence

So two ways for loading a slice into the decompressor architecture are necessary:

- If the slice can be compressed, the corresponding Sc_i will be loaded in one cycle through the adder (*add mode*).

- If not, the slice is loaded in $\lceil N/M \rceil$ cycles by serial loading, bypassing the adder (*shift mode*).

B. Compression Algorithm

As for all compression techniques, a requirement is that the initial test sequence contains don't care bits (X's) (otherwise most of the slices cannot be compressed). Anyway, when applying the test sequence to the circuit, those X's must be assigned. The number of uncompressible slices, and thus TAT, depends on the X's assignment.

The first step of the algorithm consists in initializing the first slice by assigning each X in it. The number of compressible slices depends on this initialization. For instance, in the example given in Figure 3, if the first slice S_1 is initialized to 0000 0011, no Sc_2 can be found to achieve compression towards S_2. A good assignment is to look over the next slices of the original test sequence until a specified bit is found. The corresponding X is then assigned to this value.

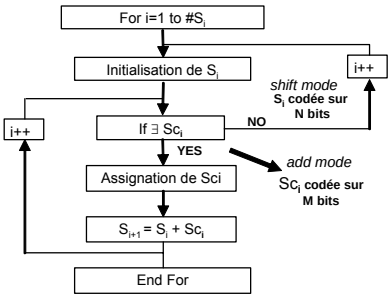

Figure 4: Compression algorithm

Then, for each Si in the test sequence, the algorithm described on Figure 4, seeks after a Sc_i such that $S_{i+1} = S_i + Sc_i$. Since S_{i+1} may contain X's, several values for Sc_i may exist. The assignment of S_{i+1} thus depends on the choice of the assignment of Sc_i which have, in turn, an influence on the possibility of assigning S_{i+1} and so on and so forth. So assigning the bits of Sc_i has an influence on the number of compressible slices since this number depends on the values in assigned original slices. For instance, in the example given in Figure 5, two choices are possible for Sc_2, (01 or 11). The first choice allows compressing until the third slice whereas the second choice leads to the sixth slice as depicted in the figure.

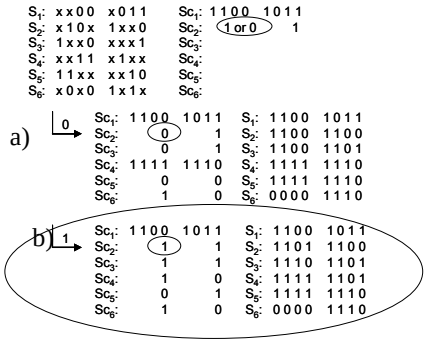

Figure 5: Depth-first search algorithm

C. Compression Results

In this section we report results on ISCAS'89 benchmarks. Two kinds of test sequences have been used for comparing the serial loading and the compression technique. For this last one, we first used a test sequence with X's, called *type 2* in Table 1. To have a fair comparison between compression and serialization, we used for serialization another test sequence which does not contain any X and thus is very shorter (called *type 3* in Table 1). All test sequences have been obtained with help of the Synopsis ATPG tool TETRAMAX [1].

			type2: compression - 217 pat,						type3: serialization - 160 pat,			
L	N	M	% X	Nsl_{P2}	original volume (bits)	comp volume (bits)	Nsl_c	Time (cycles)	Nsl_{P1}	Volume (bits)	Time (cycles)	Time %
27	8	1	82,57	5859	46872	13181	4813	14471	4320	34560	39067	62,96
		2				17670	4867	10071		34560	21787	53,78
		4				25472	5350	7121		34560	13147	45,84
7	32	2	83,19	1519	48608	16928	1056	9151	1120	35840	19207	52,36
		3				18177	1065	6737		36960	13607	50,49
		8				21992	1109	3383		35840	5767	41,34
		16				29904	1169	2443		35840	3527	30,74

Table 1: Results for the ISCAS'89 benchmark s5378, for 8 and 32 scan chains

Table 1 reports the results for the s5378 circuit, for two scan chains configuration. The 214 flip-flops are first organized in 8 scan chains of length 27, and then in 32 scan chains of length 7. Results with M varying from 5% to 50 % of N are presented. In this table, *%X* denotes the X's ratio in the test sequence, and *time (cycles)* is the number of ATE clock cycles needed to apply the whole test sequence. The last column of the table represent the gains achieved by our compression method using *type 2* test sequences over serial loading using *type 3* test sequences

It can be noticed that with the same number of ATE channels, our method provides a gain over serialization ranging from 30 to 60% in terms of test time. Other experiences have been realized benchmarks with various M and N configurations. Those experiences confirm the same behavior as with the s5378 circuit.

III. APPLICATIONS

A. *TAM-based System- on-Chip*

A SoC test architecture is proposed by the IEEE 1500 Standard [6]. It mainly consists of a Test Access Mechanism bus (TAM) which bitwidth is W_{TAM}, and wrappers around cores C_i. Each wrapper interfaces the core and the TAM bus. An ATE with WATE available channels is used to test the system. With such an architecture, reducing test time means optimizing the use of the TAM bus and the test scheduling of cores (Figure 6).

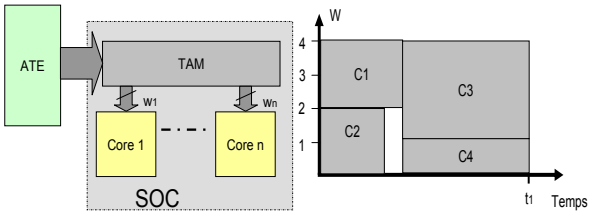

Figure 6: test architecture and scheduling without compression

The use of TDC techniques in this context allows the use of a larger TAM bus for a given number of ATE channels. This increases cores test parallelism and thus reduces test time (Figure 7).

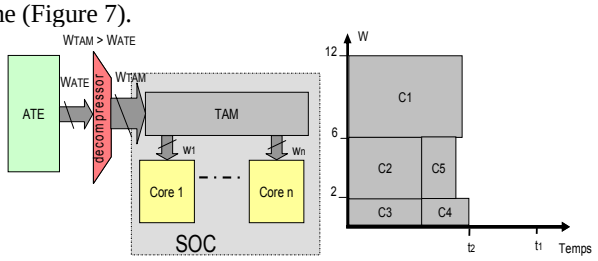

Figure 7: test architecture and scheduling with compression

Nevertheless, there are constraints bounded to the use of TDC in this context. Indeed, to obtain the best core architecture/scheduling for a given system, each core test time must be known. But using compression, test time depends on the compression ratio W_{ATE}/W_{TAM} and on test data. If two cores are tested through the same decompressor (e.g. on Figure 8-a, cores C1 and C4 are tested through the same decompressor), each time the configuration changes, a new test time for the two cores have to be recomputed. This solution is prohibitive in terms of CPU time. So Cores must be tested serially on a bus. In consequence, a multi buses structure is used, with one single decompressor per bus. Doing so, parallelism is increased and the constraint of serially testing cores on a bus is satisfied (Figure 8-b).

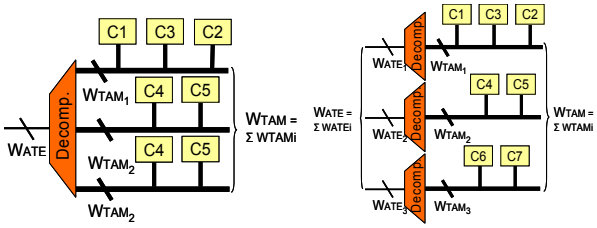

Figure 8-a & 8-b: architecture examples

We propose here a tool that provides an optimal solution in terms of test time for a given system (number of cores, number of test buses, etc …) and for a given ATE. First all the possible combinations of buses are explored. The ATE channels partition can be easily determined knowing the total W_{ATE} width and the number of lines p. First, all the possible ATE channels partition are explored(A partition is a way of writing an integer W_{ATE} as a sum of p positive integers where the order of the addends is not significant, i.e $\sum w_{i\_ATE} = W_{ATE}$ with i =1..p). The number of partitions can be easily determined knowing the total W_{ATE} width and the number of lines p. Then for each ATE partition, the algorithm explore all compatible TAM partitions such as $\sum w_{i\_TAM} = W_{TAM}$ and $w_{i\_TAM} \geq w_{i\_ATE}$. For each pair of W_{ATE} and W_{TAM} partitions, compressions ratio for each lines are fixed to w_{i_TAM} / w_{i_ATE}. Each core test time is thus defined for each line and a scheduling can be performed to obtain the best core assignment in terms of global test time. Once all pairs of W_{ATE} and W_{TAM} partitions have been explored, the optimal configuration and its associated scheduling are memorized. An external loop on the number of line p allows exploring all possible solutions. A complete architecture reducing test time is thus provided.

	Exp1	**Exp2**	**Exp3**
ATE (32 bits)		1/1/2/12/16	5/5/6/8/8
TAM (64 bits)	11/11/12/14/16		6/10/12/16/20
Time (cycles)	38596	**89892**	**42513**

Table 2: résultats pour un système de 16 cœurs ISCAS'89 [5]

Experimental results have been obtained on a SoC (defined in [5]) that contains 16cores . The first result (Exp1 in Table 2) reports test time for a 64 bits TAM and a 64 bits ATE without using compression ($W_{ATE}=W_{TAM}$). The second results (Exp2) reports test time for a 32 bits TAM and a 32 bits ATE, still without compression. The last one (Exp3) is the result provided by our tool for a 64 bits TAM and a 32 bits ATE ($W_{ATE}=W_{TAM}/2$) with the use of our TDC technique.

It can be noticed that if a 64 bits ATE is available, Exp1 solution without compression leads to the best result. But TDC has allowed to divide by two the number of ATE channels at the expense of only a 10% increase on TAT. Moreover with a 32 bits ATE, our method allows the use of a TAM of 64 bits instead of a 32 bits without compression. This point leads to an increase in cores test parallelism and thus to a gain in test time of 50%. The configuration obtained with our tool is depicted in Figure 9.

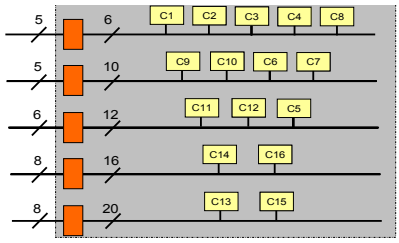

Figure 9: resulting architecture with our tool

B. Networks on Chip

A new way of designing multiple-core-based Systems-on-chip is to use a network instead of the classical communication buses. The Network-on-Chip (NoC) will transfer packets of data from a core to another (through a path of routers) just like packets are transferred on the Internet between two computers (Figure 10-a). Cores are connected to the network by wrappers specifically designed for NoC usage.

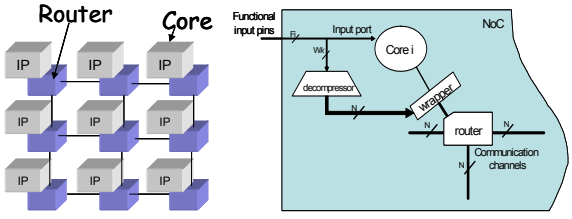

Figure 10: Decompressor Insertion in the Network

NoC specific test techniques have thus been developed (e.g. in [3]). The NoC is reused to transfer test data from ATE to cores instead of the TAM in classical bus-based SoC. With this kind of architecture, test time mainly depends on the number of test ports that can be made up according to the number of I/O pins available on the system. For Instance with a 32 bits NoC, an ATE with 64 channels and a system with 64 input pins, two 32-bits ports can be used. In this context, TDC technique allows using more test ports while keeping the number of I/O pins and the number of ATE channels constant. For instance, using the decompressor depicted in Figure 10-b, one test port may use only 4 input pins and 4 ATE channels while sending 32 bits words in the NoC. The algorithm determines the optimal ports configuration in the same way as for TAM-based systems but without exploring the TAM partitions. Indeed, in NoC-based systems, the size of the NoC is fixed (e.g. 32 bits). So for each ATE partition, the scheduling presented in [3] is performed to obtain the smallest test time. After the complete process, the optimal configuration and its associated test scheduling is obtained.

	ATE	compression	Number of ports	Time (cycles)
Exp1	**32 bits**	**no**	**1**	**36588**
Exp2	120 bits	no	5	9652
Exp3	**32 bits**	**yes**	**5**	**12853**

Table 3: results for d695 with a 32 bits NoC

For instance, with a 32 bits ATE and a system with a 32 bits NoC, compression allows using 5 test ports. Experimental results on the d695 of the ITC'02 benchmarks suite [4] are reported in Table 3. If 5 ports configuration is used without the help of TDC techniques, a 160 bits ATE is required. In this case test time is smaller but with an increase in ATE channels of 400%. We propose here a method that reaches a gain in test time of 65% without any increase in ATE costs. So, like for TAM-based Systems-on-Chip, compression allows increasing the number of test ports and thus increase test parallelism what leads to smaller test times.

IV. CONCLUSION

A new compression scheme has been presented in this paper. It is independent of test data, independent of circuits netlists and it does not required any specific tool. This method is particularly adapted to the framework of Systems-on-Chip testing. Indeed, in the Systems-on-Chip Framework, if used jointly with a scheduling tool, it allows gains in terms of test time up to50% for classical TAM-based Systems-on-Chip, and up to 65% for NoC based systems.

REFERENCES

[1] http://www.synopsys.com/products/test/tetramax_wp.htm

[2] Julien Dalmasso, Marie-Lise Flottes, Bruno Rouzeyre: Fitting ATE Channels with Scan Chains: a Comparison between a Test Data Compression Technique and Serial Loading of Scan Chains. DELTA 2006: 295-300

[3] Chunsheng Liu, Hamid Sharif, Erika Cota, Dhiraj K. Pradhan: Test Scheduling for Network-on-Chip with BIST and Precedence Constraints. ITC 2004: 1369-1378

[4] Erik Jan Marinissen, Vikram Iyengar, Krishnendu Chakrabarty: A Set of Benchmarks for Modular Testing of SOCs. ITC 2002: 519-528

[5] V. Iyengar, A. Chandra, "A Unified SOC Test Approach Based on Test Data Compression and TAM Design", DFT'03, pp: 511-518.

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