Solving the I/O Bandwidth Problem in System on a Chip Testing *

Walid Maroufi, Mounir Benabdenbi, Meryem Marzouki†
LIP6 Laboratory
Couloir 55-65, 4 Place Jussieu
75252 Paris Cedex France
Tel. (+33)1 44 27 71 23 - Fax. (+33)1 44 27 72 80
E-mail : Walid.Maroufi,Mounir Benabdenbi,Meryem Marzouki@lip6.fr

Abstract

The first part of this paper describes the control of CAS-BUS, a P1500 compatible Test Access Mechanism (TAM). Boundary Scan features are used to allow controlling of the TAM and the P1500 wrappers. The final architecture characteristics are its flexibility, scalability and reconfigurability. It also allows trade-off to optimize test time and area overhead.
The second part deals with a test pin expansion method in order to solve the bandwidth problem. The solution we propose is based on a new compression/decompression mechanism which avoid TAM performances degradation.

1 Introduction

While electronic systems are becoming more and more complex, testing capabilities face difficulties to follow the design evolution. By integrating complete systems on a single chip, the designers can answer new consumer application demands. Hence, they improve IC performances: large bandwidth, high speed, low power consumption, reduced area... However, these Systems On a Chip (SoCs) introduce new challenges. SoC design is based on the integration of core designs (DSP, microprocessor, RAM,...) provided by different companies. Testing these cores, which can be deeply embedded in the chip, can then only be done after the SoC has been manufactured. Three types of features are needed to test a SoC: test sources and sinks, a Test Access Mechanism (TAM) and core test wrappers [1]. The industry needs some standardization for SoC testing, and the IEEE P1500 working group [1] proposes to only standardize the core wrapper. The SoC integrator is then responsible for the choice of TAMs, and sources and sinks well adapted to the chip constraints. Many TAM architectures [2], [3], [4] and some test control solutions [2], [5], [6] can be found in the literature.
Some constraints may degrade the TAM performances. The one we focus on in this paper is when The TAM width is greater than the SoC test pins number. In such a case, the expansion of one or more external input and output pair of serial test pins is necessary in order to connect all test bus wires. This will obviously increase test time and then degrade test performances, since we have to expand one test pins pair into two or more serial test data chains. To cope with this problem, we propose an expansion mechanism, based on a compression/decompression method, allowing to limit test time increase. This novel compression/decompression method [7], which reaches a compression degree between 25% (minimum average) and 50% (maximum) is primarily dedicated to our architecture (CAS-bus + TAP control) but it can be adapted to other architectures. The 25% minimum average gain is completely independent from any test data correlation. The choice of other compression/decompression statistical coding methods like those based on Huffman or similar coding [8] or test vector decompression via cyclical scan chains techniques [9] was studied, but their dependence on test data correlation represents an important drawback.
We have presented in [10] a scalable and reconfigurable TAM named CAS-BUS. This TAM can be controlled by different means but our choice has been to use the well known IEEE 1149.1 features to provide not only the TAM's control but also the wrappers control at the same time [11]. The first part of this paper briefly presents the TAM architecture and its control. The second part focuses on the compression/decompression mechanism dedicated to test pins expansion.

2 The CAS-BUS TAM

The CAS BUS (figure 1) is a TAM which main function is to provide access to embedded cores whatever the wrapper is. This TAM is made up of two main elements:
- **a bus** consisting of N wires,
- **a Core Access Switch (CAS)** (figure 2)
Each CAS selects from the N wires of the bus (e_i) the P ones that will be applied to the core wrapper inputs (o_i). It also connects the P outputs of the wrapper (i_i) to the CAS outputs (s_i). The CAS is composed by a Switch

*This work has been partly supported by MEDEA AT403-SMT Project
†Contact author

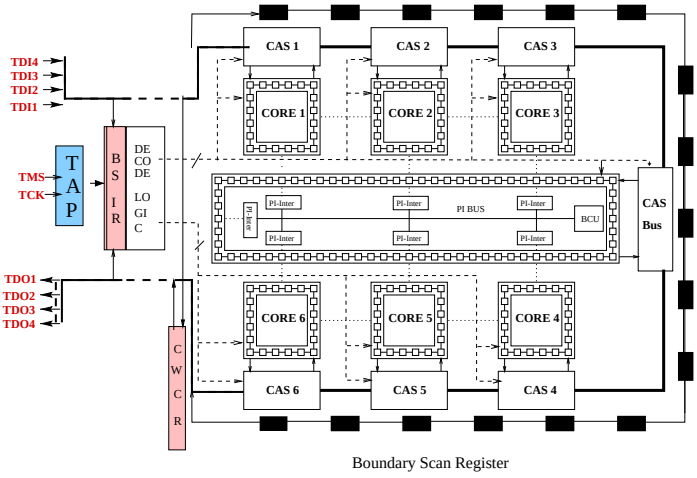

Figure 1: SoC Test architecture

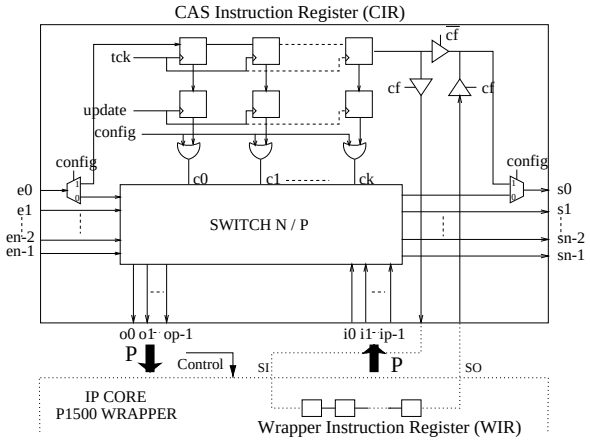

Figure 2: CAS architecture

and a CAS Instruction Register (CIR). It is controlled by several signals. Signal c_f can connect the CIR to the Wrapper Instruction Register (WIR). Signals c_i control the Switch.
Thanks to the CAS Instruction Register (CIR), three functional modes are available.

- the **configuration** mode: All c_i are set to 1. A k bit word is shifted in the CIR through the first wire of the bus. In this mode, all CIRs are serially connected. Each loaded word corresponds to a N/P switch scheme. Depending on c_f value, the Wrapper Instruction Register (WIR) can be serially connected to the CIR.

- the **bypass** mode: The N inputs e_i go through the Switch to the N outputs s_i.

- the **test** mode: When the loaded instruction is updated, the routing is made through the Switch. While the P wires o_i (i_i) are correctly multiplexed to (from) the wrapper, the N-P remaining ones are bypassed. The core is then ready for test.

Controlled by the k bit word of the CIR, the Switch is in charge of routing the test stimuli and the test responses to/from the core wrapper. The Switch is made up of one or more decoders and of some logic gates. The decoder controls the multiplexing of the N/P I/Os. By selecting P wires from the N ones existing at the Switch inputs, the decoder must be able to address all the possible N/P combinations.
The number of combinations as well as the decoder area obviously depend on N and P values. Some results obtained for different values of N and P can be found in [11].
This CAS-BUS TAM architecture is independent from any industrial proprietary SoC architecture. It can be used with any kind of wrappers and thus gives the SoC integrator more freedom for designing its SoC test architecture. The CAS-BUS is scalable, flexible and reconfigurable. More details on this architecture can be found in [10] [11].

3 SoC Test Architecture and Control

The control of this TAM architecture must be simple without degrading the TAM advantages. Two options have been considered: defining an ad hoc test control mechanism or reusing a standard control architecture. The decision to reuse the IEEE 1149.1 Test Access Port (TAP) naturally came to manage our TAM functionalities. This solution fits well with our SoC test architecture, since it allows TAM control, wrapper control and usual Boundary Scan functionalities at the same time. The test control must allow to manage three mandatory test steps: the configuration of the different core wrappers, the configuration of all the CASes, and the application and the analysis of the test vectors.
Two ways of wrapper configuration are possible. The first one is to connect the CAS instruction Register (CIR) to the Wrapper Instruction Register (WIR), thus the WIR configuration is made together with the CIR configuration. The other possibility is to let the SoC test integrator define a specific test strategy to configure the WIR if this one is not connected to the CIR.
Since it has been decided to allow both options, we have extended the normal Boundary Scan architecture with a CAS-Wrapper Coupling Register (CWCR). The CWCR width is equal to the number of CASes. Each bit of this register corresponds to the value of the configuration signal c_f of each CAS (figure 2).
Three new instructions are needed to control the three mandatory test steps. They have been defined as Boundary Scan optional instructions.

- the **CAS-Wrapper_Coupling** instruction

  It defines which CIR/WIR pairs must be connected in order to configure at the same time a core wrapper and the corresponding CAS. TDI1 and TDO1 are

connected to the CWCR. c_f values are shifted in the CWCR to indicate which WIRs will be connected to the CIRs.

- the **CAS_CONFIG** instruction

  This instruction configures all the CASes, together with the wrappers selected with the instruction CAS-Wrapper_Coupling. The config signal is set to 1 to provide access to the CIR of each CAS. TDI1 and TDO1 are respectively connected to input e0 of the first CAS and to output s0 of the last one. The CIR and the WIR are then loaded with the appropriate values, putting each CAS in the correct scheme and each wrapper in the chosen mode.

- the **CAS_TEST** instruction

  When loaded, this instruction allows the test vectors to be applied to the different cores and the test stimuli to be propagated to the SoC test outputs. All TDIs and TDOs are connected to the I/Os of the CAS BUS. The set of test vectors can be concurrently shifted in to the IP cores inputs and the responses shifted out from the IP outputs.

In our modified TAP architecture, TDI and TDO is a bus. This TDI/TDO bus width may vary from 1 (normal Boundary Scan architecture) to N (width of the CAS bus). This width can be chosen by the SoC integrator, depending on SoC pins availability.
The simplest case occurs when TDI/TDO bus width is equal to N. When the TDI/TDO bus width must be lower than N, then it is necessary to expand the TDI inputs to the CAS inputs. Such an expansion mechanism, designed in order to reduce test performance degradation, is described in section 4.
Using the IEEE 1149.1 control part with scalable TDI/TDO lets the CAS-BUS TAM keep its main benefits and adds to it flexibility.
The global architecture is not only scalable and flexible but also dynamically reconfigurable. By correctly configuring the CASes, the test programmer can choose during each test session which core scan chains must be serialized in order to optimize their total length. The goal is to have for each of the N wires the same test length. If not, when testing the SoC, don't care stimuli must be applied at TDI inputs to complete the balance between scan chain lengths. Another advantage is that WIRs whose mode remains unchanged between two test configurations can be bypassed during the CAS CONFIG phase. Using this reconfigurability aspect, test time can be saved.
Compared to other approaches described in the literature, the global architecture, TAM plus control features, is very flexible and easily controlled using the well known TAP controller. It allows the concurrent application of test stimuli to many wrapped cores through a scalable bus. Even if they are too different to allow any comparison, architectures like HTAP [5] or TLA [6] have only one TDI/TDO couple and do not allow to test many cores at the same time.
In the CAS-BUS architecture, Boundary Scan I/Os are used not only to control the CASes and the core wrappers, but also to apply test stimuli to cores inputs, contrarily to approaches like [12]. Indeed, even if this architecture includes Boundary Scan features at SoC level like our architecture, the TDI/TDO wires are only used to control the global SoC test, the TAM being directly connected to the SoC primary I/Os.
The CAS-BUS architecture is simple, allows trade-off and provides a complete solution for testing systems on a chip.

4 Compression/Decompression expansion method

The method we propose is a lossless statistical one which treats 4 bit-length samples of the test data. The choice of 4 bit-length samples will be explained later.

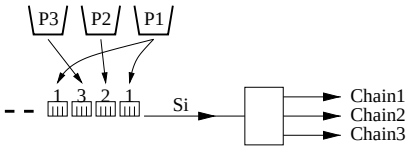

Figure 3: Example of expansion

To ease the explanation of this method, let's take the expansion example shown in figure 3. It consists of three scan chains which must be accessed through only one serial input (**si**). P1, P2 and P3 respectively represent the global length of test patterns for chain1, chain2 and chain3. We assume a cyclic transfer of the 4 bit-length samples coming on **si** to the three chains. Without any compression of test data, it is clear that the total test time needed to transfer all test patterns is equal to $(P1 + P2 + P3)$ cycles. This represents a large test time, knowing that if we had 3 serial inputs, the total test time would have been reduced to max(P1,P2,P3) cycles (parallel testing of the three chains).

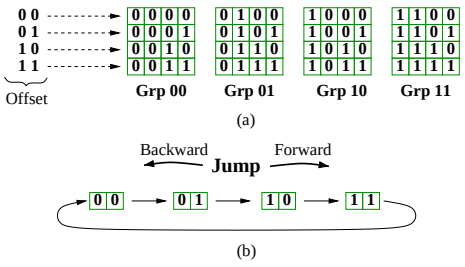

Figure 4: Groups example

For 4 bit-length samples, the number of possible combinations is equal to 16. We arbitrary classify the 16 combina-

tions in 4 groups of 4 combinations each (example: figure 4-a).
Within one specific group, each combination is normally coded using 2 bits (offset of the combination in the group). Groups are also coded 2 using bits, but we consider another way to select them. Indeed, we consider that they are cyclically chained (figure 4-b), so there are three possibilities to change from one group to another:

- One forward jump in the cyclic group chain.
- One backward jump.
- Two backward (or forward) jumps.

Thus, the idea is to start from one chosen group (for example group 00) and to code samples using 2 bits. If any group modification is necessary, a dedicated signal (for example **Grp_chg**) must indicate it. When the **Grp_chg** signal is active, the value at the input **si** indicates the direction of the jump (1: forward jump, 0: backward jump). In case of two jumps, the signal **Grp_chg** must stay active during two clock cycles while the value at the input **si** must be also stable (at 0 or at 1).
The gain for each 4bit-length sample can be estimated as follows: (i) case of no jump: 2 bits (2 cycles) for the coding $->$ 50% gain, (ii) case of one jump: 2 bits (2 cycles) for the coding and one cycle for the jump indication $->$ 25% gain, (iii) case of two jumps: 2 bits (2 cycles) for the coding and two cycles for the jumps indication $->$ 0% gain.
When there is no special correlation in a test data amount, the percentage of jumps can be statically seen as: 25% of no jumps, 25% of one forward jump, 25% of one backward jump and 25% of two jumps. So the global average gain can be calculated as : (0.25 x 50%) + (0.25 x 25%) + (0.25 x 25%) = 25%. Of course this gain is not static and essentially depends on the number of jumps within a test process. The maximum gain is reached when no jump is necessary and is equal to 50%.
One way to limit the number of jumps, is to modify the distribution of the 16 main combinations among the 4 groups. For each test process, an optimal distribution can be found.
To limit the number of possible jumps, the number of groups must be small. Suppose now that we use a 6 bit-length samples, the number of possible combinations will be 64. Distributed among four groups (for example), we will obtain 16 combination per group: 4 bit for coding offsets. In case of no jumps, the gain will be equal to (6-4/6)=33% which is less than the gain obtained for 4 bit-length samples. When distributed among more groups, 16 for example, offsets will be coded using 2 bits but the number of jumps will be very important. Suppose for example that we must make a jump of 8 groups, so 8 clock cycles must be added to the offset 2 cycles: then we have 10 cycles to code a 6 bit-length sample (-66% gain!). The same problems occur for larger samples (8, 10, ...), thus we decided to use four bit-length samples.

5 Integration within the CAS-bus/TAP architecture

As said in the introduction, the compression/ decompression method we present is first dedicated for the CAS-bus architecture controlled using the 1149.1 TAP presented in section 2. The serial input is obviously TDI, and in order to avoid additional signals we decide to use the TMS signal to indicate a jump. An expansion/decompression module is added to the architecture as well as an MISR from which a signature can be shifted out through TDO (figure 5). In

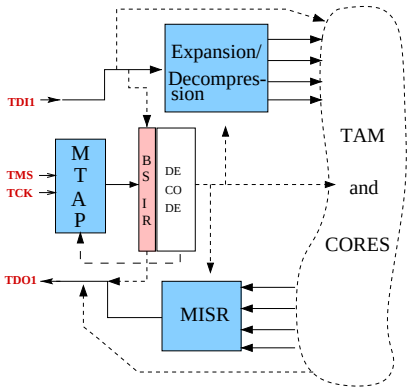

Figure 5: CAS-bus/TAP architecture with expansion/decompression module

addition to the three new instructions presented in section 2, a fourth instruction called **DECOMPRESS_test** replaces **CAS_test** when transferring test vectors. This instruction connects the TDI serial input to be expanded to the expansion/decompression module and the TDO output to the MISR.
It is important to note that only one clock cycle is allowed to indicate a jump (2 cycles for two jumps). Thus, using the TMS signal to indicate group jumps with a classic standard TAP controller is impossible. Indeed, when TMS is activated within a test vector transfer, it is impossible to go back to the **shift-DR** state before 3 clock cycles, which does not satisfy constraints.
The solution we propose is to use a modified TAP which we call MTAP. This TAP behaves exactly like a classic standard 1149.1 controller except when a **DECOMPRESS_test** is shifted in. In that case, two additional states (**ONE-jump** and **TWO-jumps**) are added to the data register control part of the TAP. This is automatically done when the instruction register decoder decodes the **DECOMPRESS_test** instruction.
The main advantage of MTAP is that the constraint of one clock cycle for one jump is satisfied. Indeed when TMS signal is activated within a test vector transfer, MTAP moves to the state **ONE-jump** (the direction of the jump depends on the value of TDI). If another jump is necessary, TMS must stay at one and MTAP moves to the state **TWO-jumps**, otherwise it goes back to a shift

state: **shift-DR**.
Figure 6 shows an expansion/decompression sequence of one TDI input to three serial chains. **CHi-VALj** represents a 4 bit-length sample which is part to chain i. **CHi-CDj** indicates the compressed code of **CHi-VALj**. The four groups are named A, B, C and D and the register **Current-group** is initialized to B. Once three codes (ex: CH1-CD1,CH2-CD1 and CH3-CD1) are shifted in, the decompressed values (CH1-VAL1,CH2-VAL1 and CH3-VAL1) are shifted out through the three chains. It is important to note that the **Pause-DR-bis** and **Shift-DR-bis** signals replace the initial **Pause-DR** and **Shift-DR** signals for the three chains control. However, the initial signals (**Pause-DR** and **Shift-DR**) still control the expansion/decompression module serial internal chain.

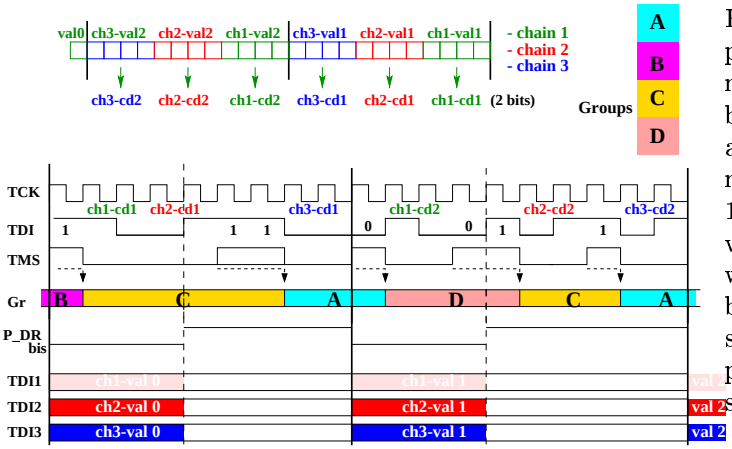

Figure 6: Expansion/Decompression sequence

Since 12 clock cycles are involved to shift three 4 bit-length samples without any compression/decompression process, it is simple to see the gain obtained for each sequence: 3 cycles for CHi-VAL1 and 2 cycles for CHi-VAL2.

5.1 Implementation

In order to check the feasibility of the method, we implemented a cycle precise synthesizable description of the decompression module, the MISR, the MTAP and all the IEEE 1149.1 remaining parts. The modular description was synthesized with the SYNOPSYS design analyzer. Simulations coincide well with expected results.
In the synthesized description, groups combinations can be modified during the step of global configuration. A (16x4)bits internal register is chained with CAS instruction ones. Of course, it is possible to do that differently, depending on the designer constraints.
The decompression module was synthesized in a static configuration, which means that the number of serial output was fixed (4 outputs). However, it is possible to imagine a reconfigurable module with a scalable number of derived chains. In its static version, the module contains 40 flip-flops (without counting the groups flip-flops) and less than 500 combinational gates, which doesn't represent an important area overhead compared to the SOC global area.

5.2 Example

In order to experiment the method with a real test example and considering the lack of real SOC application availability, we decided to apply it to the test patterns of a real circuit designed in our laboratory. The circuit, named PCIDDC, is a 200K transistor, Network interface Component for PCI bus including 4 scan chains with a 1028 total scan length.
Using our method, we easily reach 35% of compression rate by randomly modifying the distribution of combinations within groups. A software module is dedicated to that and the 35% rate was obtained after few processing minutes.
For the same patterns, we calculated the percentage of appearance of each combination in order to prepare a Huffman coding: (i) with 4 bit-length samples : all the 16 combinations have a close appearance degree (between 5.3% and 8%), (ii) with 6 bit-length samples : all the 64 combinations have a close appearance degree (between 1% and 1.9%). This means that correlation between samples is very loose and no important compression can be reached with a Huffman coding. Obviously, for all the methods based on similar coding, the problem remains the same, since they all necessitate a good correlation between samples. This kind of problems are avoided with the expansion/decompression method detailed in this paper.

5.3 Special cases

Two special cases can occur with the presented expansion/decompression method:

The first case arises when no expansion is needed (configuration: one serial input and one internal chain). In such a case, two test clocks are necessary: one slow clock for the compressed patterns (Tester) and a fast clock, for the decompressed chain. The same technique is used in [8]. Note that this is not necessary in case of an expansion, since the time needed to input compressed codes is at least equal to 2N cycles (N is the number of target internal chains, N >= 2), which is enough to shift in parallel 4 bit-length samples.
The second case arises when more than one serial input is available. In such a case, a simple rule is to minimize the number of chains to be connected to the expansion/decompression module. So, for N serial inputs and M internal chains, we can connect (N-1) inputs to (N-1) chains and use the last input to expand the remaining (M-(N-1)) chains. The distribution of chains on the direct or expanded serial inputs must be a result of a global tradeoff taking into account test chains length and corresponding test data amount. Using more than one decompression module is possible, but this necessitates more than one MTAP mechanism.

6 Conclusion

The global architecture presented in this paper is both P1500 compliant at core level and 1149.1 compliant at SoC level. It offers a complete solution for testing SoCs. This global test architecture is flexible and scalable, allowing test designers and test programmers to optimize test time and area overhead. Trade-off regarding the choice of Nmax and the CAS configuration can be made.
A compression/decompression method can be integrated within the CAS-bus/TAP architecture. Even if it does not give spectacular compression performances with correlated test patterns like Huffman/similar or run-length coding methods, it is very useful with non correlated test patterns. Compression degree varies between 25% (minimum average) and 50% (maximum). We propose this method to enhance test performances in term of test time when expanding a test serial input to more than one SOC internal test chain. An expansion/decompression sequence example has been presented. A simple modification has been introduced in the IEEE1149.1 TAP controlling the CAS-bus architecture. The modified TAP (MTAP) behaves exactly as a classic TAP, except in case of an expansion/decompression process. The functionality of the global architecture: CAS-bus + MTAP + Decompression Module + MISR has been validated at gate level. The method has also been tested on the real test patterns of a circuit made in our laboratory (PCIDDC: a Network Interface Component) and 35% of compression degree is reached. The use of other compression methods was studied for the same patterns, but due to their feeble correlation, compression degrees were not significant.

References

[1] E.J. Marinissen, Y. Zorian, R. Kapur, T.Taylor, and L.Whetsel. Towards a standard for embedded core test: An example. In *IEEE International Test Conference (ITC)*, pages 616–627, Atlantic City, NJ, September 1999.

[2] L. Whetsel. Addressable test ports an approach to testing embedded cores. In *IEEE International Test Conference (ITC)*, pages 1055–1064, Atlantic City, NJ, September 1999.

[3] E. J. Marinissen and al. A structured and scalable mechanism for test access to embedded reusable cores. In *International Test Conference*, Washington, DC, October 1998.

[4] P. Varma and S. Bhatia. A structured test re-use methodology for core-based system chips. In *International Test Conference*, Washington, DC, October 1998.

[5] D. Bhattacharya. Hierarchical test acces architecture for embedded cores in an integrated circuit. In *IEEE VLSI Test Symposium (VTS)*, pages 8–14, Dana Point, CA, April 1998.

[6] L. Whetsel. An ieee 1149.1 based test acces architecture for ics with embedded cores. In *IEEE International Test Conference (ITC)*, pages 69–78, Washington, DC, November 1997.

[7] W. Maroufi. A new compression/decompression method for non correlated test patterns: Application to test pins expansion. In *IEEE European Test Workshop (ETW)*, Cascais, Portugal, May 2000.

[8] A. Jas, J. Ghosh-Dastidar, and N. A. Touba. Scan vector compression/decompression using statistical coding. In *17th IEEE VLSI Test Symposium*, San Diego, California, April 1999.

[9] A. Jas and N A. Touba. Test vector decompression via cyclical scan chains and its application to testing core-based designs. In *International test conference*, pages 458–464, 1998.

[10] M. Benabdenbi, W. Maroufi, and M. Marzouki. Cas-bus: A scalable and reconfigurable test acces mechanism for systems on a chip. In *IEEE Design Automation and Test in Europe (DATE)*, pages 141–145, Paris, France, March 2000.

[11] W. Maroufi, M. Benabdenbi, and M. Marzouki. Controlling the cas-bus tam with ieee 1149.1 tap: A solution for systems on a chip testing. In *4th IEEE International Workshop on Testing Embedded Core-Based System-Chips*, Montreal, Quebec, Canada, May 2000.

[12] J. van Beers and Harry van Herten. Test features of a core-based co-processor array for video applications. In *International Test Conference*, pages 638–647, Atlantic city, NJ, September 1999.