CAS-BUS: a Test Access Mechanism and a Toolbox Environment for Core-based System Chip Testing

Mounir Benabdenbi Walid Maroufi* Meryem Marzouki
LIP6 Laboratory
Couloir 55-65, 4 Place Jussieu, 75252 Paris Cedex 05, France
Mounir.Benabdenbi@lip6.fr Meryem.Marzouki@lip6.fr

Abstract

As System on a Chip (SoC) testing faces new challenges, some new test architectures must be developed. This paper describes a Test Access Mechanism (TAM) named CAS-BUS that solves some of the new problems the test industry has to deal with. This TAM is scalable, flexible and dynamically reconfigurable. The CAS-BUS architecture is compatible with the IEEE P1500 standard proposal in its current state of development, and is controlled by Boundary Scan features.

This basic CAS-BUS architecture has been extended with two independent variants. The first extension has been designed in order to manage SoC made up with both wrapped cores and non wrapped cores with Boundray Scan features. The second deals with a test pin expansion method in order to solve the I/O bandwidth problem. The proposed solution is based on a new compression/decompression mechanism which provides significant results in case of non correlated test patterns processing. This solution avoids TAM performance degradation.

These test architectures are based on the CAS-BUS TAM and allow trade-offs to optimize both test time and area overhead. A tool-box environment is provided, in order to automatically generate the needed component to build the chosen SoC test architecture.

1 Introduction and State of the art

With the ever growing integration density, current and future ICs include more and more capabilities. Complete systems on a single die, chips containing numerous IP cores are commonplace today. These cores can be DSPs, CPUs, memories... and can be provided to the system integrator by different companies. The reuse of these IP cores significantly reduces the time to market of such Systems on a Chip (SoCs) [1].

*Now with Nortel Networks, Ottawa

However, while geometry shrinking and design reuse allow impressive gains, SoC testing faces new set of problems and has become one of the bottlenecks of the IC industry progress. Cores coming from different companies with different functionalities, various Design for Testability (DFT) features and delivered at different levels of description (soft, firm or hard) harden the work of the system test integrator.

Solving these problems needs new types of test architectures, able to manage the test of up to 100 million transistors cores while allowing the high fault coverage required before signing-off a design to manufacturing. Moreover, highly standardized solutions are needed in such a context. The efforts of an IEEE working group [2] have resulted in the IEEE P1500 standard proposal of a SoC test architecture, which main elements are, in its current development status:

- **Test Sources** to generate the core test stimuli and **Test Sinks** to compare the test responses to the expected ones.
- A **Test Access Mechanism (TAM)** in charge of transporting test data between sources, cores and sinks.
- A **Core Test Wrapper** [3] which is the interface between the embedded core and the TAM. Through different modes, it provides test functions at the core terminals.

The major effort of the working group focuses on the wrapper standardization problem, while source, sink and TAM design is left to the system designer. So new test architectures must be developed to access core I/Os from the SoC primary I/Os, some cores being deeply embedded in the circuit. TAM architectures can be based on the use of the system bus [4] or on a specific test bus [5], [6]. A detailed study of existing proposed architectures can be found in [7].

The SoC test integrator should be provided with different TAM architectures, in order to decide which one is best adapted to a given core-based SoC. Various criteria may be taken into account, such as:

- The level or core access and test preparation. The core may be wrapped or not, or include Boundary Scan features easing its accessibility (TAPed cores). It may also include different internal DFT capabilities.

- The complexity of the whole test control.

- The amount of test patterns to be applied to the circuit. With the integration density growth comes the increase of the number of patterns needed to test the whole SoC and test patterns at core level must be transferred at SoC level [8]. This obviously leads to a test time increase.

- The packaging of the SoC. The number of IC pins is sometimes a constraint since it defines the number of pins dedicated to test capabilities.

- The area overhead induced by test feature inclusion.

- The external test equipment that will be used to test the SoC. Depending on the Automated Test Equipment (ATE) capabilities (memory, bandwidth...), the SoC test integrator will select the best TAM architecture for the SoC design.

These constraints lead the test engineer to make multiple trade-offs regarding the choice of the appropriate TAM and its characteristics.

Among these constraints, testing time is a key issue. Methods and algorithms to optimize TAM width have been presented in [9] and in [7]. This optimization has been extended to include power and routing constraints ([10]). To complete these approaches, a wrapper/TAM co-optimization has been developed and presented in [11]. These approaches reduce test application time but are not sufficient if not associated with test scheduling. A test framework including a set of design algorithms, dealing with test scheduling and test resource placement has been recently proposed in [12].

Some TAMs are designed to access cores with standard or proprietary wrappers ([5], [6], [13]). We have developed a scalable and reconfigurable TAM architecture, named **CAS-BUS** [14], which falls into this category. This TAM is detailed in section 2, as well as the developed tools to generate and use such an architecture.

The main benefit of this TAM over the other approaches is that it is dynamically reconfigurable. As others it allows test time optimizations during its design by optimizing the bus width. However what is different is that it provides test time optimization capability during the test pattern application step by changing the test data routing scheme, from one test session to the other.

Some others TAMs access TAPed cores, where the Boundary Scan features of the core are used instead of a wrapper to access the core I/Os ([15], [16], [17]).

But, to our knowledge, very few of the existing TAMs deals with both wrapped cores and TAPed cores integrated in the same SoC. However, a system integrator may need to integrate some TAPed cores together with P1500 wrapped cores, when these TAPed cores have primarily been designed as stand-alone chips. This need may arise in order to keep the SoC time-to-market as short as possible, thus avoiding the redesign of already TAPed cores to suppress their boundary scan features and replace them with P1500 wrappers. In addition, a TAPed core cannot simply be wrapped on top of IEEE 1149.1 features, because of area overhead as well as boundary scan rules regarding hierarchical global test control. In [18] the problem began to be adressed but the approach is limited to a TAM width equal to one (one TDI/TDO couple). To cope with these real-world needs, we have extended our basic CAS-BUS TAM in such a way that SoC populated with both wrapped cores and TAPed cores may be easily tested. This extension, which has been first presented in [19], is now detailed, together with the presentation of its design environment, in section 3 of this paper.

Another real-world concern deals with available SoC I/O bandwidth, when the TAM width is larger than the number of SoC test pins. In this case, the expansion of one or more external I/O pair of serial test pins into parallel internal signals would be mandatory, in order to connect all test bus wires. Otherwise, this would obviously increase test time and degrade test performances of the SoC. To overcome this constraint we developed a test pin expansion mechanism [20]. It is based on a compression/decompression method, allowing to limit test time increase. This compression/decompression method is primarily dedicated to our architecture (CAS-BUS TAM), however it can be adapted to other architectures. The main benefit of this method is that the gain in test time is completely independent from any test data correlation. Many approaches based on statistical coding ([21], [22]) take advantage of the probabilistic distribution of test data. The approach we propose here is aimed to be used when test data samples have the same probability of occurence. This method and its implementation tools are detailed in section 4 of this paper.

The control of the basic CAS-BUS TAM architecture and its two instances (the one considering TAPed cores inclusion and the one solving the bandwidth problem) has also been addressed [23], and is described in the corresponding sections of the paper. Some comments on the peculiarities of the control process of each instance are provided in section 5. Experimental results for the implemented architectures are presented in section 6

2 The Basic CAS-BUS TAM

The CAS-BUS (figure 1) is a TAM which main function is to provide test access to embedded cores whatever the wrapper is. As an example, the wrappers described in this paper are IEEE P1500 wrappers in the current state of development of the standard proposal, since the CAS-BUS architecture is compliant with this ongoing standard.

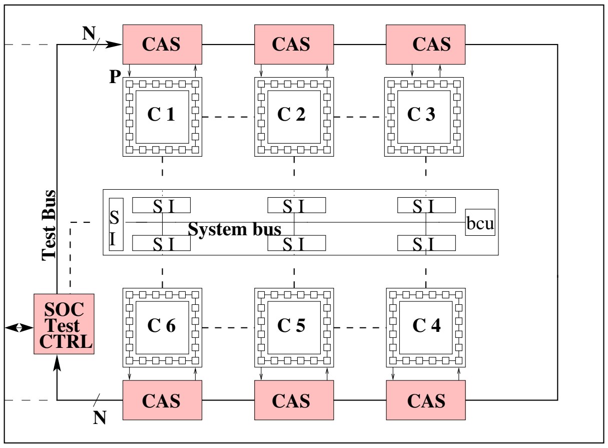


Figure 1. Basic CAS-BUS TAM architecture

2.1 Architecture

This TAM is made up of two main elements:

- A **Core Access Switch (CAS)** (figure 2), which is a simple programmable router connected to each testable core through the wrapper at its test terminals,

- A **test bus** which is a set of wires transporting test data through the SoC and connecting CASes to each other.

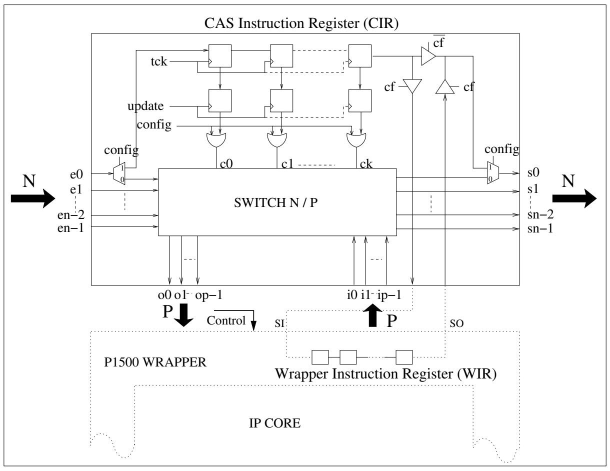


Figure 2. CAS router architecture

Let N be the width of the test bus, and P the number of test pins for a given core. N is greater or equal to 1 and P is lower or equal to N. The CAS chooses among the N wires composing the test bus the P wires which will be connected to the test pins of the core. P depends on the core test method:

- For **scannable** cores, P is the number of integrated scan chains (figure 3-a),

- For **BISTed** cores, P is generally equal to 1 (figure 3-b),

- For cores tested using **external** sources and sinks, P depends on the nature of these source and sink, e.g. P=1 when the source is a simple LFSR and the sink a simple MISR (figure 3-c),

- For **hierarchical** cores, we consider that internal cores can be CASed, and in this configuration P is equal to the width of the internal test bus (figure 3-d).

All test control signals, either for the CAS or for the testable cores, are connected to a central SoC test controller which is in charge of synchronizing test data and control. 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). Unselected inputs (e_i) are bypassed to the outputs (s_i).

The CAS is composed by a **switch** and a **CAS Instruction Register (CIR)**. It is controlled by several signals. For example signal c_f can connect the CIR to the Wrapper Instruction Register (WIR) and signals c_i control the switch.

Controlled by the CIR, the switch is in charge of routing the test stimuli and the test responses to/from the core wrapper. Two kinds of switch implementation are possible:

- (a) The switch can be made up of N decoders (one per input wire) and some logic gates (figure 4-a).

- (b) The switch can alternatively be made up of only one decoder and some logic gates (figure 4-b).

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.

To avoid timing problems, a flip flop cell is inserted before each CAS output (s_i). This kind of cell is not present at (o_i) and (i_i) I/Os since they are connected to the internal wrapped core registers. The critical delay may appear during the updating of the routing scheme when using a switch with one decoder and with great values of N and P. In this case delay grows with the decoder area. However this kind of switch is not intended to be used for large values of N and P.

The number of combinations as well as the switch area obviously depend on N and P values and on the kind of implementation. Results obtained for different values of N and P can be found in section 6.1.

2.2 Control

The control of the CAS-BUS architecture (figure 5) 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,

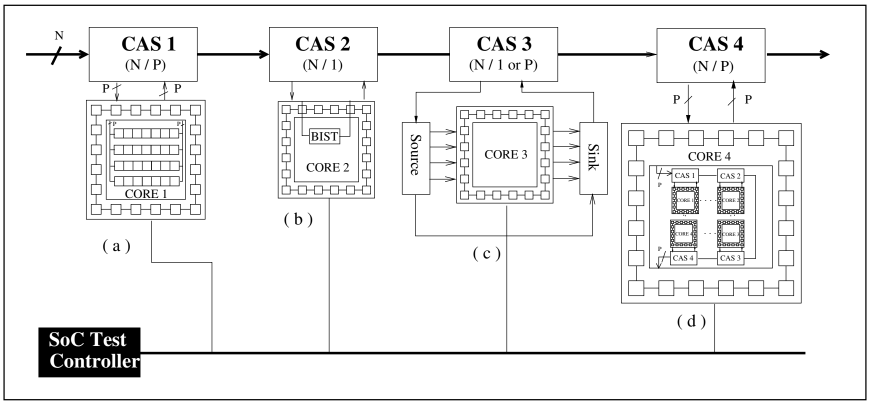


Figure 3. Test types supported by the CAS-BUS

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
- The application and the analysis of the test vectors.

The normal Boundary Scan architecture has been extended 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 for each CAS.

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 (figure 6-a)

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

- The **CAS_CONFIG** instruction (figure 6-b)

It configures all the CASes, together with the wrappers selected with the instruction CAS-Wrapper_Coupling. The config signals provide access to the CIR of each CAS. TDI1 and TDO1 are respectively connected to first input of the first CAS and to first output 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 (figure 6-c)

When loaded, it 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 into the IP core 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 CASBUS). This width can be chosen by the SoC integrator, depending on SoC pin availability.

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.

2.3 Developed Tools

Two independent software modules have been developed to ease the integration of this TAM in the SoC architecture.

2.3.1 CAS Generator

The SoC test designer first has to decide the width of the test bus (N) and the number of switched wires for each CAS (P).

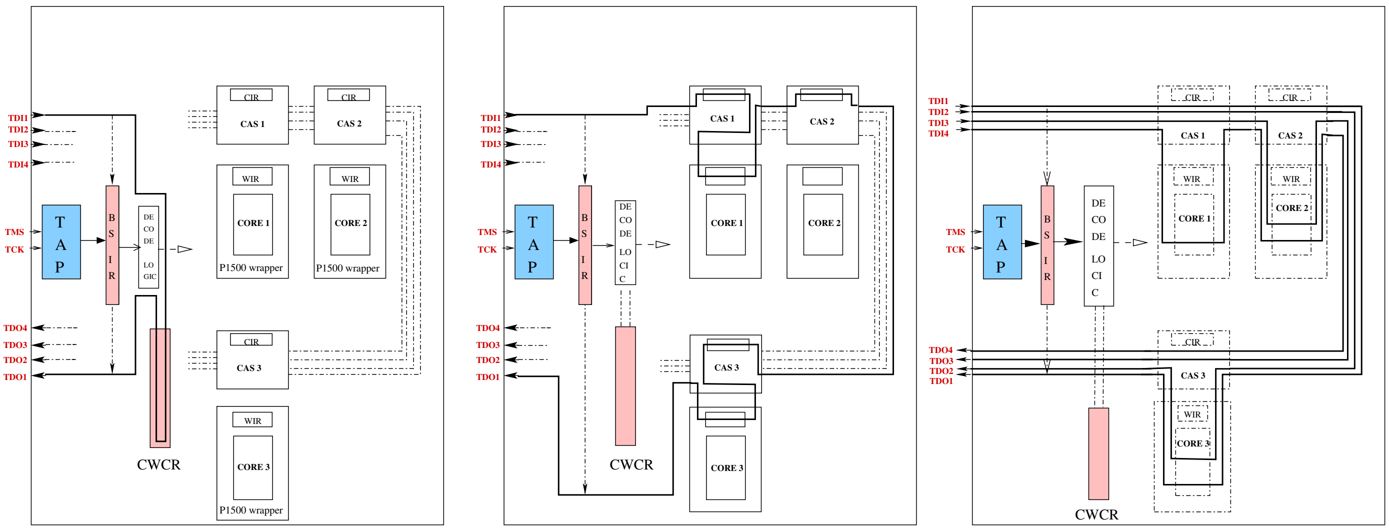
In order to reduce the number of combinations and thus the width of the instruction register, the following heuristic has been defined (figure 2): *When an input e_i is switched to an output o_j, the corresponding i_j CAS input is switched to the s_i output.*

The use of this heuristic obviously limits the number of combinations needed and reduces the control complexity. Then we can completely construct a test bus wire path from an *e* input to an *s* output with the same control word ($c_0...c_k$).

Thus, for *m* combinations the width *k* of the instruction register can be calculated using the formula:

$$k = \lceil log_2(m) \rceil$$

The CAS generator developed is based on this heuristic and takes as parameters the N and P values and the kind of switch implementation. It provides a VHDL description



(a) CAS−WRAPPER COUPLING (b) CAS CONFIG (c) CAS TEST

Figure 6. The three new mandatory instructions

of the chosen CAS, which can be synthesized with a any synthesis tool. This generator is written in the C language.

2.3.2 Wrapper Generator

A wrapper generator has also been developed. The software, designed in C language, takes as input the VHDL description of a core I/Os. It generates the corresponding wrapped core in the VHDL format. The generated wrapper is as close as possible to the IEEE P1500 wrapper, as the standard is still under development. The design of the wrapper is based on public information that can be found in [2]. In order to be easily updated, the generator is designed as a collection of specialized modules, and is based on the use of a cell library. The cells are those described by the standardization group on its web site. Some choices had to be made since some features are not completely defined yet. Currently, the generator takes as parameters the kind of boundary cell (basic or shift/update cell) and a file where are described the characteristics of the core INOUT pins. We currently use this file to solve core INOUT pins problems. This file will be replaced by the Core Test Language (CTL [2]) file that will be provided with the IP core, when the standard will be completed and adopted.

2.4 Benefits and Comparison to Other Approaches

The CAS-BUS TAM architecture has multiple advantages over other approaches described in the literature, due to its flexible, scalable, reconfigurable and dynamic aspects:

- It supports different core test methods (Scan, BIST, parallel test),

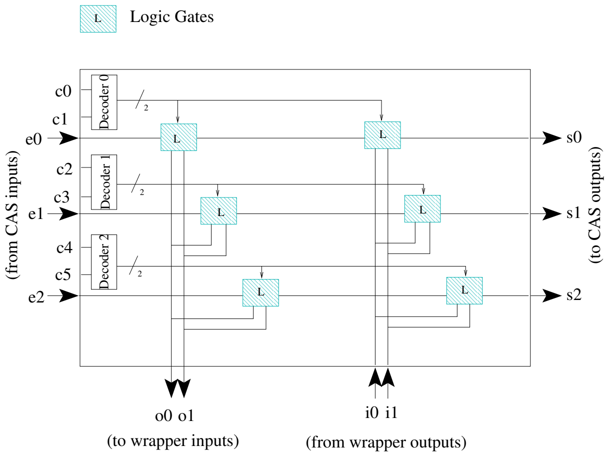
- It can provide parallel test access to many cores, depending on N and P. If N is limited, the cores can be partially or completely daisy chained. Whatever the routing scheme is, many wrappers can be active simultaneously.

- It allows the test of hierarchical cores without degrading performances in terms of reconfigurability and flexibility,

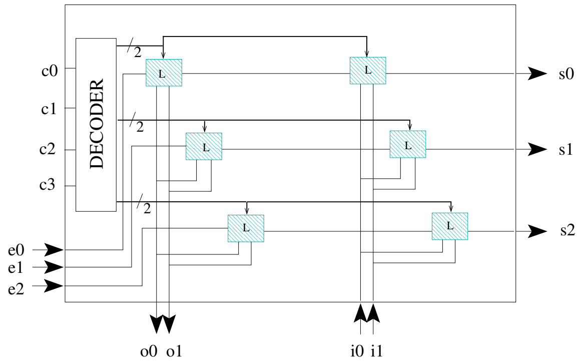
- It can easily be reconfigured, even during test sessions, in order to optimize test performances. A good collaboration between the test designer and the test programmer leads to a good trade-off between test overhead and test time. For example, in case of scanned cores, the test programmer can balance the length of the scan chains within the test programs, in order to reduce the test time. In the same way, SoC interconnect test time can be optimized when adopting a good configuration of the test chains,

- It allows maintenance test, by testing some embedded cores while others are in normal functioning mode. This is very useful when, e.g., an embedded memory test is periodically required.

Moreover, the CAS-BUS has the advantage of being independent from any industrial proprietary SoC architecture. In addition, we propose an architecture where the TAM can be used with any kind of wrappers (either proprietary or standard to be defined, e.g. the P1500 wrapper in its current status), contrarily to approaches like [5], where the TAM and the wrapper are closely merged, leaving few freedom of decision to the system integrator. On the con-



(a) Switch with many decoders



(b) Switch with one decoder

Figure 4. Two types of switch architectures

trary, the CAS-BUS TAM eases the SoC test architecture design by using plug-and-play CAS modules.

Like the Adressable Test Port architecture ([13]), the CAS-BUS TAM is only intended to provide test access to the embedded cores, independently of the wrapper configuration. However many differences appear between the two architecures. Among these differences we can see that in the CAS-BUS TAM the same bus is used to input and output data and many different cores can be tested at the same time. The CASes are configured to determine the routing scheme contrarily to the Adressable Test Ports that are addressed for enabling purpose. However the CASes do not provide capabilities such as test generation and test compaction allowed within the Adressable Test Ports.

With its control features based on the IEEE 1149.1 standard, the CAS-BUS TAM 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

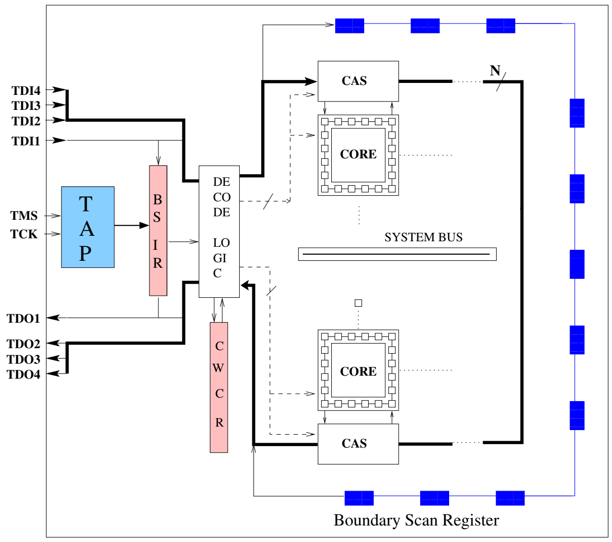


Figure 5. CAS-BUS Test Control

to allow any comparison, architectures like HTAP [15] or TLA [17] 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 [24]. 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.

3 Extension of the Architecture to TAPed Cores Testing

The basic test architecture presented in section 2 can be used when the cores within the SoC are wrapped cores. Since TAPed cores are not wrapped cores, they can only be accessed through their test I/Os (TMS, TDI, TDO, TCK and $TRST$, if present). Unlike wrapped cores, these cores include a TAP controller. Obviously this needs to define a hierarchical test architecture able to drive these TAP controllers. Moreover, configuring the TAPed cores in the chosen mode cannot be done in the same way as wrapped cores. Direct access to the wrapper shift/update instruction register (WIR) can be enabled, while the instruction register within the TAPed core can only be accessed through the TAP finite state machine. Thus TAPed cores and wrapped cores cannot be configured serially in the same test session.

Our goal being to test both wrapped and TAPed cores within the same test session and with the same TAM, some improvements must be made on the CAS-BUS architecture in order to solve the problems induced by the TAP

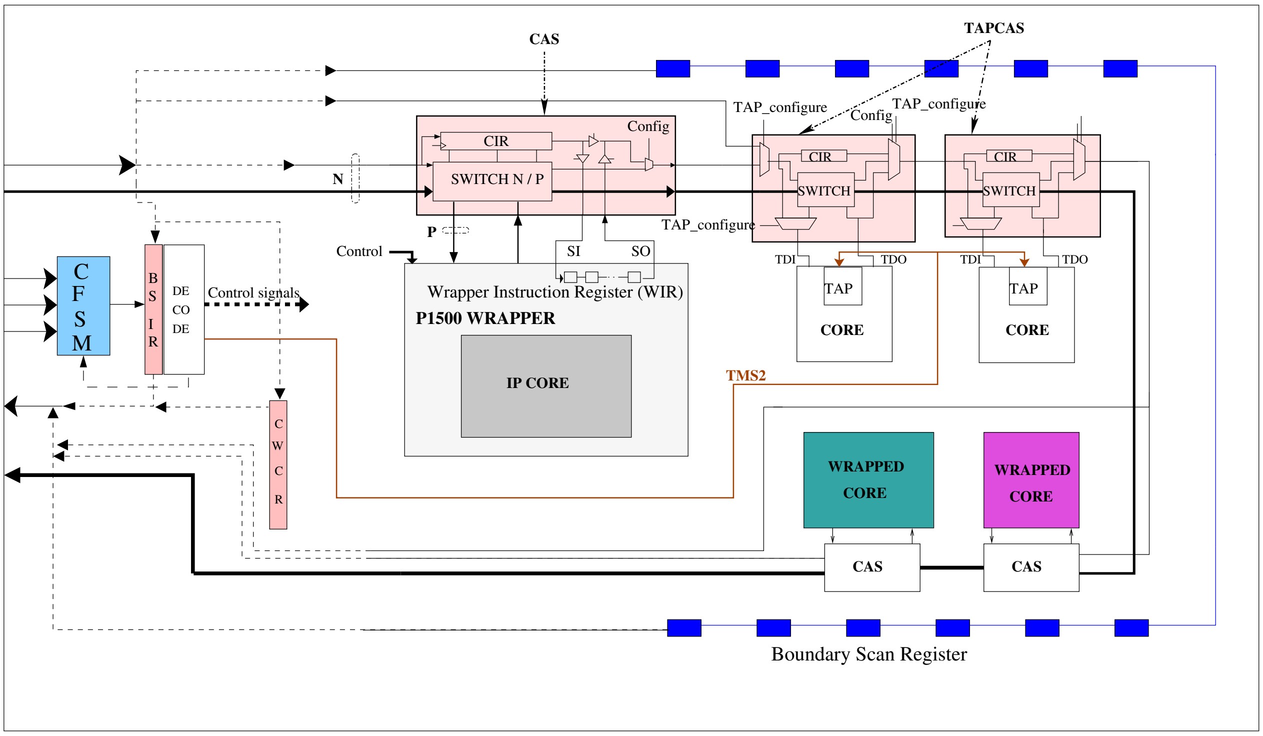


Figure 7. SoC Test architecture for TAPed and wrapped cores

controllers, while avoiding any IEEE 1149.1 standard rule violation.

The two main improvements consist of defining on the one hand a special Core Access Switch for TAPed cores and of modifying on the other hand the central TAP controller of the CAS-BUS architecture in order to manage the hierarchical aspect of the global control.

3.1 Core Access Switch for TAPed Cores

Since TAPed cores cannot be serially configured with the wrappers, a new Boundary Scan instruction named **TAP_CONFIG** has been added, which enables the configuration of the TAPed Cores. The configuration of the wrappers is still made by the CAS_CONFIG instruction. Figure 7 shows a SoC example containing both kind of cores. This TAP_CONFIG instruction serially connects CASes of TAPed Cores (TAPCASes) one to the other and connects each TAPCAS to its associated core. It also connects $TDI1$ and $TDO1$ respectively to the first and the last TAPCAS.

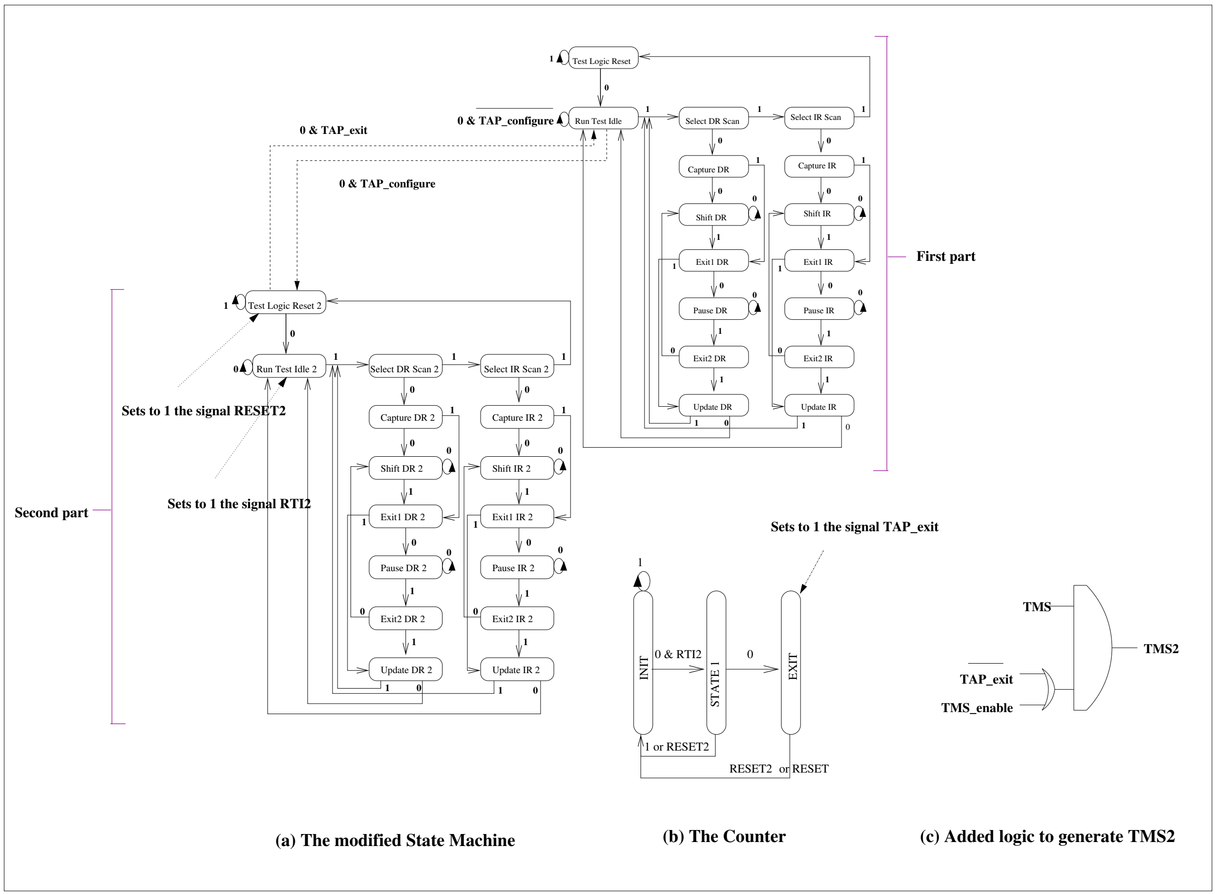
The TAPCAS is slightly different from the normal CAS. The CIR and the switch remain the same, the tristates disappear and multiplexers are added. However, two kinds of TAPCAS architectures are needed, depending on the placement of the TAPed core within the SoC. In the global chaining of all the CASes, if the TAPCAS follows a CAS then the TAPCAS must have a multiplexer at its inputs. In the other cases this multiplexer is not needed.

Obviously, for the TAPCAS, P is equal to 1: the TAPCAS is connected to TDI/TDO pair of the TAPed core. Depending on the boundary scan instruction, test data at TAPCAS inputs can be routed differently. The different multiplexers will route these data to:

- The **TAPCAS instruction register (CIR)**. This is done with the **CAS_CONFIG** instruction. The switch is configured like all the other switches.

- The **internal instruction registers of the TAPed cores**. In that case the **TAP_CONFIG** instruction must be loaded. A configuration word is shifted in the internal instruction registers of the TAPed cores, these registers being daisy-chained. The data skip the switch in the TAPCAS and are directly applied to the TDI input of the TAPed core. The different operations like shifting, updating, etc., are executed thanks to the TMS2 signal that drives the internal TAP controller (figure 8). This signal TMS2, described in the next section, is generated from the central TAP controller. Once the internal instruction registers are updated, the TAPed cores are then in the correct test mode (bypass, INTEST, etc.) and ready for test vectors application.

- The **internal data registers of the TAPed cores**. When the **CAS_TEST** instruction is loaded, test vectors are routed from one of the N wires of the TAPCAS in-



(a) The modified State Machine (b) The Counter (c) Added logic to generate TMS2

Figure 8. SoC Test architecture controller

puts to the TDI input, through the switch. The test vectors are shifted in the internal test data registers (boundary scan register, bypass register...). The responses to the stimuli are then shifted out to the next core or to the SoC outputs. Here also the shift/update operations are controlled through TMS2.

3.2 The New Central TAP Controller (CFSM)

The new central TAP controller (figure 8) is made up of a 1149.1 like TAP Finite State Machine (FSM) and a counter.

This Central FSM (CFSM) is in charge of controlling the global architecture as a classical boundary scan feature but must also drive the different Internal Finite State Machines (IFSM) of the TAPed cores. To do that, it was decided to add to the usual 16 states of a TAP, a second set of 16 new states corresponding to the states of the TAPed cores. Except Test logic 2 and Run Test Idle 2, the added states do not drive any signal and only are transition states. Depending on the instruction loaded, TMS will control the CFSM and the IFSMs or only the CFSM.

The test instructions to be loaded in the Boundary Scan instruction register should respect the following chronological sequence:

- 1. **CAS_WRAPPER_COUPLING** to define which CIRs and WIRs will be connected during the configuration step. This instruction operates only on CASes and wrapped cores.

- 2. **CAS_CONFIG** to configure the CASes and the TAPCAS. The routing scheme of the overall switches is defined.

- 3. **TAP_CONFIG** to configure the test mode of the TAPed cores.

- 4. **CAS_TEST** to apply test stimuli and shift out the test responses. The TAPed cores and the wrapped cores are tested concurrently through the same bus.

The following subsections describe more in details what happens when the TAP_CONFIG and the CAS_TEST instructions are loaded.

3.2.1 TAP_CONFIG Instruction Effect

This instruction is needed to shift values in the instruction registers of the TAPed cores.

When this instruction is loaded and updated, the signal TAP_configure is set to 1. Run Test Idle state (figure 8-a) is left to enter in the second part of the CFSM in the Test Logic Reset 2 state. This state sets to 1 the RESET2 signal that sets the counter (figure 8-b) in the state INIT. The counter is initialized, the signal TAP_exit is set to 0. TMS2 (figure 8-c) will then have the same value as TMS. To synchronize the CFSM and the IFSM, which states must be equivalent to those of the second part of the CFSM, TMS must be held at 1 during 5 cycles to reset the IFSM.

When entering in the Run Test Idle 2 state, the signal RTI2 is set to 1, Reset2 is set to 0 and if TMS is held at 0 during 2 cycles, the counter is put in the state EXIT. The TAP_exit signal is set to 1. This sets TMS2 to 0 and the IFSMs are no more controlled by TMS. The IFSMs are and stay in the Run Test Idle state. The counter is used not only to enable and disable IFSMs control but also to leave the second part of the CFSM.

If TMS is not held at 0 during 2 cycles, when the CFSM is in the Run Test Idle 2 state, the counter stays in the INIT state. The IFSMs are controlled by TMS and their current state is equivalent to the current state of the second part of the CFSM. The instruction words can be shifted in the TAPed core as they would be if placed on a boundary scan board.

3.2.2 CAS_TEST Instruction Effect

This instruction is used to shift in, apply and shift out test vectors to and from all the cores within the SoC. With the right configuration of the CASes and the TAPCAS, all the test data registers are connected as multiple scan chains. Internal test data registers of the TAPed cores (boundary scan or bypass registers) can be considered as internal scan chains of the P1500 wrapped cores. When updated this CAS_TEST instruction sets to 1 the signal TMS_enable and sets to 0 the signal TAP_configure. TMS controls the IFSMs through TMS2 (figure 8-c). During this test step the current state of the CFSM cannot be one of those of the second part.

When leaving the Update IR state (figure 8-a), the next state must be Run Test Idle. The configuration step of the TAPed cores having left the IFSMs in the Run Test Idle state, the synchronization of the CFSM and the IFSMs is done thanks to this state. This way, the current state of all the finite state machines of the SoC being exactly the same, shifting and applying test vectors within wrapped and TAPed cores can be made at the same time.

The IFSMs remain controlled by the CFSM until a new instruction is loaded in the Boundary Scan Instruction Register (BS IR) of the SoC. This new instruction will set to 0 the signal TMS_enable. This is equivalent to setting TMS2 to 0 and the IFSMs are left in the state Run Test Idle. However, while shifting this new instruction in the BS IR, unknown values are concurrently shifted in the internal instruction registers of the TAPed cores. This is not important because the test responses have already been shifted out from the internal test data registers and because the next test session will reset these instruction registers. The next TAP configuration step will load these registers with the right values.

3.3 Benefits and limitations

The main benefit of this new CAS-BUS architecture is that TAPed cores do not need to be tested apart from the other wrapped cores using added test resources, hence decreasing the test area overhead. TAPed cores can be tested with the CAS-BUS TAM taking advantage of the possible optimizations provided by this TAM. Test time can still be optimized through correct configuration of all CASes and TAPCASes.

With this architecture it is also possible to test only TAPed cores with the TAP_CONFIG instruction. The cores can be tested as stand-alone chips on a boundary scan board, controlled by the CFSM, because direct test access is provided to their I/Os, skipping the wrapped cores. However BIST testing can not be processed within these TAPed cores. During the TAP_CONFIG step the second part of the CFSM is active. If the RUN_BIST instruction is loaded in the TAPed core, to execute the self test, the current state must be Run Test Idle 2 during many cycles, with TMS held at 0. With this CFSM, this cannot be done. In that case, after TMS is held at 0 during two cycles, the next state is Run Test Idle, in the first part of the CFSM, and then the IFSMs stop being controlled by the CFSM. If really needed, a solution could be to add a register that would be concurrently loaded with the internal instruction registers of the TAPed cores. If the RUN_BIST instruction is detected in this new register, the current state remains in the Run test Idle 2 state without exiting to the main part of the CFSM.

As for the previous CAS-BUS architecture, interconnect testing can be performed, between wrapped cores and TAPed cores, assuming that the EXTEST instructions are loaded.

The CAS generator has been extended and a new parameter has been added. The user can also generate a VHDL description of the TAPCAS with the desired kind of switch, with or without the input multiplexer. Some results are discussed in section 6.1

This extension of the CAS-BUS TAM architecture does not degrade the characteristics of the previous, basic architecture, while it offers a solution for testing SoCs with TAPed cores. If the SoC does not have any TAPed cores included, the integrator should obviously prefer the basic CAS-BUS architecture.

4 Addressing the I/O Bandwidth Issue

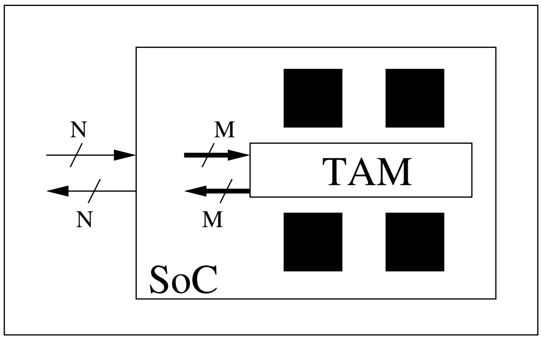


Figure 9. Problem: N < M

Some constraints may degrade the TAM performances. The one we focus on in this paper arises when the TAM width is greater than the SoC test pins number (figure 9). 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 pin pair into two or more serial test data chains (figure 10).

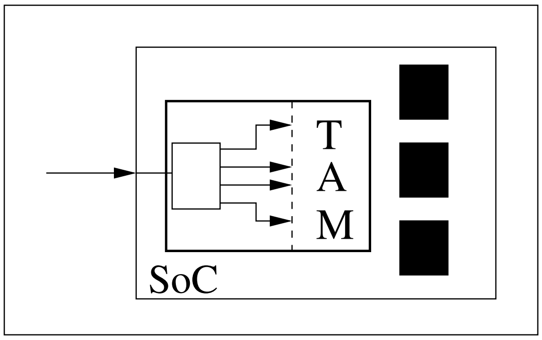


Figure 10. Expansion

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 is primarily dedicated to our CAS-BUS architecture but it can be adapted to other architectures. The compression gain reached with this method is completely independent from test data correlation. The choice of other compression/decompression statistical coding methods like those based on Huffman or similar coding [25] or test vector decompression via cyclical scan chains techniques [26] was studied, but their dependence on test data correlation represents an important drawback.

4.1 Theory

The method we propose is lossless and treats n bit-length samples of the test data. The choice of the optimal bit-length value will be explained later.

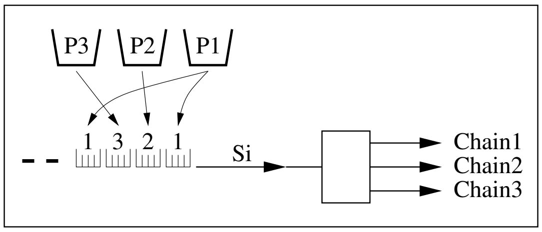


Figure 11. Example of decompression/expansion

To ease the explanation of this method, let us take the expansion example shown in figure 11, with 4 bit-length samples (n=4). 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).

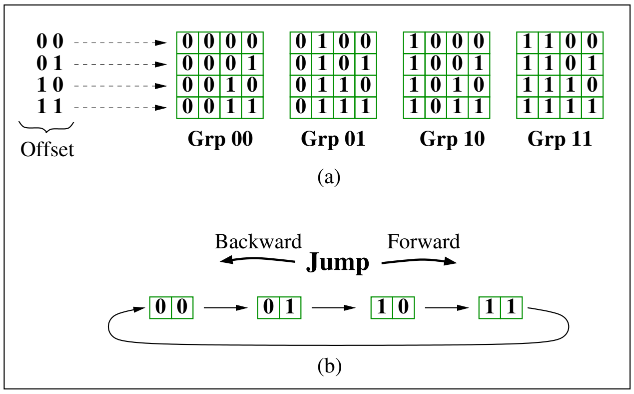


Figure 12. Groups example

For 4 bit-length samples, the number of possible combinations is equal to 16. We arbitrary classify the 16 combinations in 4 groups of 4 combinations each. Figure 12-a shows a possible example of such classification.

Within one specific group, each combination is normally coded using 2 bits (offset of the combination in the group). Groups are also coded using 2 bits, but we consider another way to select them. Indeed, we consider that they are cyclically chained (figure 12-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 the test data, the percentage of jumps can be statically evaluated 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%) + (0.25 x 0%)= 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, ...).

4.2 Integration in the CAS-BUS Architecture

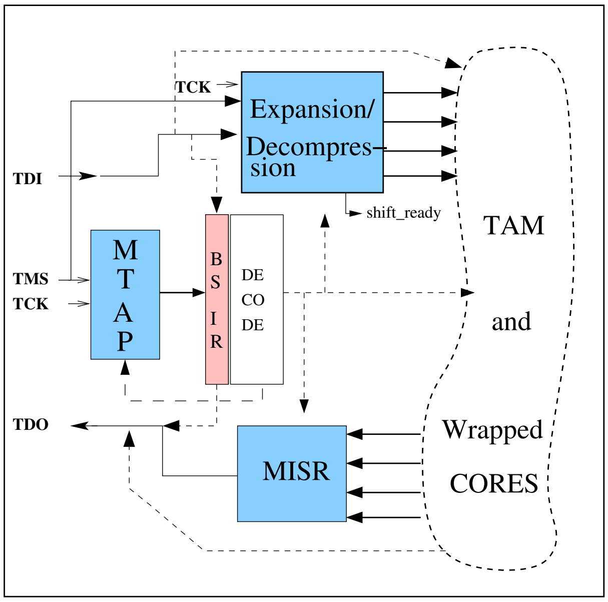


Figure 13. CAS-BUS/TAP architecture with expansion/decompression module

As said in the introduction, the compression-decompression/expansion method we present is first dedicated for the CAS-BUS architecture controlled using the 1149.1 TAP. This method has been integrated considering 4 bit-length samples. The serial input is obviously TDI, and in order to avoid additional signals we decide to use the TMS signal to indicate a jump (**Grp_chg**). 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 13).

In addition to the three new instructions presented in section 2.2, 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.

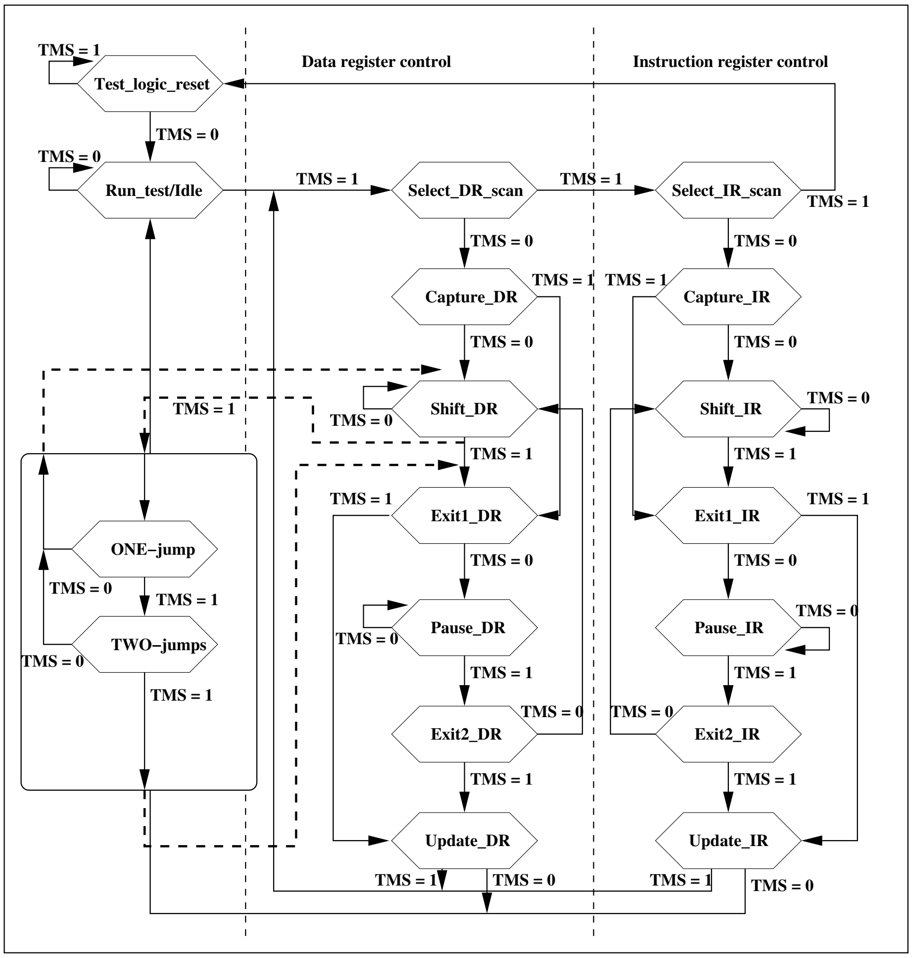


Figure 14. Modified TAP state machine

The solution we propose is to use a modified TAP which we call MTAP (figure 14). 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**.

The expansion module takes as inputs TDI, TMS, TCK and control signals generated from the MTAP controller. It outputs the N chains corresponding to the TAM width, and a control signal (**shift_ready**) specifying that decompressed data are ready to be applied to the TAM. This module includes four bit-length registers and a table containing the four group samples. As we have N chains, N registers are needed to store successively the decompressed samples. Coded samples and jump informations are shifted in, decompressed and affected to the correct register, continuously, without waiting delay. This is possible since TMS is directly connected to the module. This process is stopped if we are not in one of the states (**ONE-jump**, **TWO-jumps** and **shift-DR**). When the N registers are filled the module asserts the signal **shift_ready**. Decompressed and expanded data can then be shifted out and be applied to the TAM.

The main advantage of this implementation is that TMS is both used for controlling the MTAP and for giving jump informations to the decompression/expansion module.

Figure 15 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-**

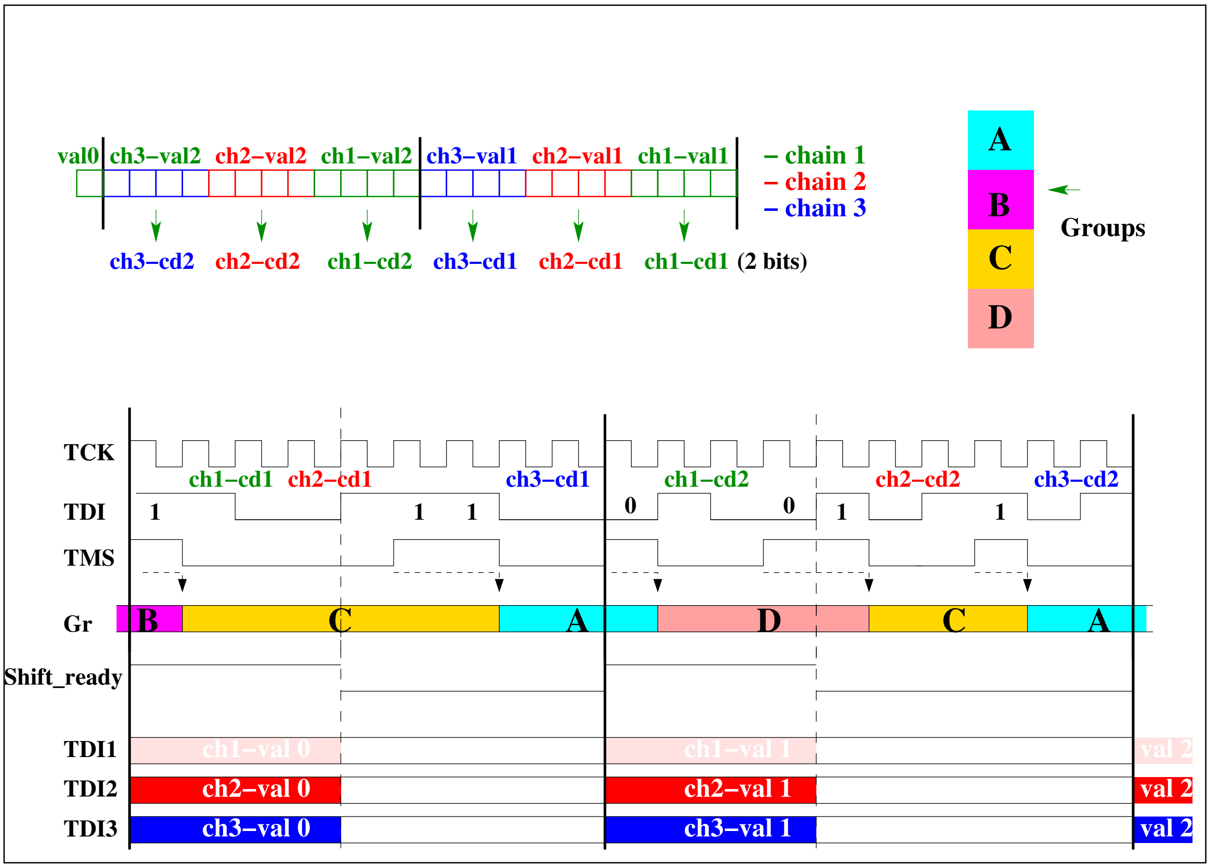


Figure 15. Expansion/Decompression sequence

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.

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

4.3 Special Cases

In this section we present two special cases which we think are the most important ones.

First special case corresponds to no needed expansion (one serial input and one internal chain). This is still possible, but two test clocks are necessary: one slow clock for the compressed patterns (tester) and the second, a fast clock, for the decompressed chain (figure 16). The same technique is used in [25]. 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 being the number of target internal chains, with N >= 2), which is enough to shift in parallel 4 bit-length samples.

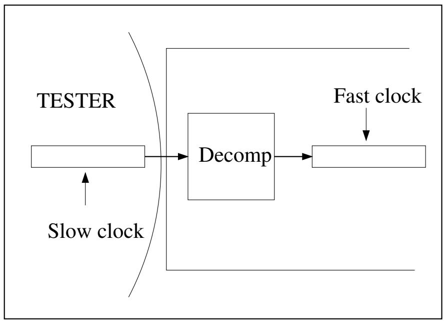


Figure 16. Case of no expansion

Second special case corresponds to using the method when more than one serial input are 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 trade-off taking into account test chain length and relative test data amount. Using more than one decompression module is possible, but this necessitates more than one MTAP mechanism.

4.4 Developed Tools

A compression software module has been developed. It first serializes the N test patterns at its inputs in one test pattern, then it makes its compression and finally include the values giving the jump directions. The compression is made according to the partitioning of the 4 groups of 4 bits. To enhance the compression degree we decided to slightly modify the proposed theory. We affect the two bits coming on TDI not to the offset value, as previously described, but to the place of the 4 bits pattern in the group. This change gives us complete freedom to partition the 16 combinations of 4 bits in 4 groups. A software module processes the serialized test pattern and gives the corresponding optimized partitioning of the 16 samples. For a given test pattern, the optimization is made in order to reduce the global number of jumps. The overall processing time needed to serialize, to calculate the best partitioning and to compress the N patterns is less than one minute. Obviously processing a test pattern in order to calculate the best partitioning corresponds to the study of the natural correlation between samples, and then use this correlation to save test time. Even if the proposed method has been developed for patterns with loose correlation, this optimization provide an overall gain on the global test time.

However when using the CAS-BUS TAM with the expansion/decompression module, before each test pattern processing, the test engineer must initialize the four groups with the optimized partition. This requires 64 extra cycles, which is very few compared to the global test length.

An application of this software to a real example is presented in section 6.2.

4.5 Discussion on sample length

The length of the sample data obviously influences the compression rate. If n is the sample's length, 2^a the number of groups and 2^b the number of samples within a group, we have n=a+b. The average compression gain is given by the formula (not detailed in this paper):

gain (%)= 100 x (1-((b+2^a/4)/(a+b)))

The optimal gain value is obtained with b=1 and a=2. Thus the optimal average gain is obtained with 3 bit-length samples. In this case the average gain is equal to 33% and the maximum gain is equal to 66%.

The 33% minimum average gain is completely independent from any test data correlation.

A version of the compression software, processing 3 bit-length samples (optimal value), has been developed. In this case the processing time is less than a second and the test engineer must initialize the four groups with 24 extra cycles only.

5 Comments on the Proposed Instances of the CAS-BUS TAM

Depending on the SoC design constraints, a choice must be made between the two test architecture instances. If the SoC is made up of TAPed and wrapped cores, then the SoC integrator should use the test architecture presented in section 3, controlled by the CFSM. If few SoC test pins are available, the architecture described in section 4, controlled by the MTAP, well fits the test integration. Merging the two architectures into a single one enabling both test pin expansion process and TAPed core testing is currently under study. As they have been defined till now, the CFSM and MTAP controllers cannot be merged into a single controller because the TAP controllers of the TAPed cores do not include the two states *ONE-jump* and *TWO-jumps*. Thus, when the instruction DECOMPRESS_test is loaded, when expanding and shifting test data, the current state of the MTAP cannot be the same as the ones of the internal controllers of the TAPed cores. This leads to define a new test controller able to drive, in the same SoC, test pin expansion modules, the CAS-BUS TAM, wrapped cores and TAPed cores.

6 Experimental results

To validate the three CAS-BUS architectures and the associated tools, simulations and synthesis have been done. Considering the lack of SoC application availability, the complete integration of the global architectures in a real SoC have not been done yet.

6.1 Architecture synthesis

6.1.1 CAS and TAPCAS generation

In our experiments, we have simulated the different features of the CAS-BUS architecture. We also have synthesized various configurations of CASes using the Synopsys synthesis tool [27]. Some results are presented in table 1. CAS (a) corresponds to a CAS which switch is implemented by N decoders and CAS (b) a CAS with a switch including only one decoder. The CAS Instruction Register width (k) and the area in terms of transistors (tr) are presented. A trade-off must be made between CAS (a) and CAS (b) depending on the area constraints and the scan cycles overhead needed for CAS configuration. For high values of N and P (great number of combinations) CAS (b) cannot be used because the area grows quickly with N and P.

These two architectures allows the SoC integrator to make a trade-off between the total *k* bits word control and the area overhead induced by the decoder size. In the first

		CAS (a)		CAS (b)		TAPCAS (a)		TAPCAS (b)	
N	P	k	tr	k	tr	k	tr	k	tr
5	1	5	696	3	626	5	684	3	614
5	3	10	1.224	6	2.026	–	–	–	–
5	4	15	1.636	7	2.634	–	–	–	–
8	1	8	1.084	4	912	8	1.072	4	900
8	3	16	1.952	9	7.286	–	–	–	–
8	4	24	2.548	11	39.990	–	–	–	–
8	6	24	2.948	15	>100.000	–	–	–	–
10	1	10	1.338	4	1.044	10	1.326	4	1.032
10	4	30	3.236	13	>100.000	–	–	–	–

Table 1. CAS synthesis results

case, the total k bit word is longer but the total number of gates is reduced. In the second case, synthesizing one decoder for the whole N inputs requires more gates but fewer control bits. A trade-off can be done to optimize the CIRs length and the area overhead. However when summing up all CIRs and WIRs length, we found out that it is far smaller than a core scan chain length. In the same way, the area overhead is not significant when compared to a core area.

Concerning the architecture dealing with TAPed cores testing, we can see in table 1 that TAPCASes (b) (one decoder per switch) are more interesting than TAPCASes (a) (N decoders per switch) since they are smaller and need fewer control bits. The area overhead induced for TAPed cores testing is not significant since the TAPCASes are very small and the new TAP controller is implemented with 1264 transistors. However configuring TAPed cores, in comparison with wrapped cores, needs 27 extra scan cycles (loading the TAP_CONFIG instruction, entering and leaving the second part of the central controller).

6.1.2 Wrapper generation

Some experiments have been made with the wrapper generator, using the AMD2901 microprocessor as a core. The wrapped core has been simulated and synthesized with ALLIANCE, a home-designed CAD suite [28]. Although the obtained wrapper area seems a very large overhead with the chosen core example, the absolute area values are very reasonable: the area of one boundary cell is equal to 98 transistors (this value may slightly vary depending on the kind of cell), and the area of the 3-bits Wrapper instruction Register (WIR) is equal to 240 transistors.

6.2 Implementation of the compression method

To check the feasibility of the method, we implemented a synthesizable description of the decompression/expansion module, the MISR, the MTAP and all the IEEE 1149.1 remaining parts. The modular descriptions was synthesized with the Synopsys synthesis tool [27]. Simulations coincide well with expected results.

The decompression/expansion module implemented deals with 4 bit-length samples and is available in generic VHDL format. An update of this module, processing 3 bit samples is under development. Its area depends on the TAM width (N). For N=3 the module's area is equal to 10488 transistors, 12842 transistors for N=10 and 31528 transistors for N=100. Another advantage of the presented method over some coding/decoding approaches is that it requires few hardware to be implemented.

In order to experiment the compression method with a real test example, 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 [29].

Using our method, when using 4 bit-length samples, we easily reach 25% of compression rate with a random distribution of combinations within groups. When using 3 bit-length samples we reach 35% of compression rate. Calculating the optimized distribution adds 2% to this compression rate. For the same patterns, we calculated the percentage of appearance of each combination in order to prepare a Huffman coding:

(i) with 3 bit-length samples: all the 8 combinations have a close appearance degree (between 10% and 20%).

(ii) with 4 bit-length samples: all the 16 combinations have a close appearance degree (between 5.3% and 8%).

(iii) 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.

7 Conclusion

The basic CAS-BUS TAM described in this paper is flexible, scalable and dynamically reconfigurable. It allows multiple trade-off regarding the choice of Nmax, the number of TDI/TDO couples, the kind of switch implementation and the CASes configurations. The architecture is both IEEE P1500-compliant (in its current development status) at core level and 1149.1-compliant at SoC level. 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. The control of the global architecture is easy through a simple test access port.

Two instances of SoC test architecture based on the use of the CAS-BUS TAM have been presented in this paper.

The first one offers a complete solution for testing both wrapped and TAPed cores within a SoC, concurrently if needed. With the same test access mechanism, wrapped and TAPed cores can be tested, thanks to a hierarchical test control. In order to manage the TAPed cores testing, new CASes have been designed and the central TAP controller has been modified. The area overhead induced by the TAPed core testing is not significant. The SoC integrator has the choice of using this architecture or the previous CAS-BUS TAM depending on the presence of TAPed cores in the SoC.

The second instance has been developed to integrate a new compression-decompression/expansion method within the CAS-BUS TAM architecture. Even if it does not provide 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 33% (minimum average) and 66% (maximum). Particularly, 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 IEEE 1149.1 TAP controlling the CAS-BUS architecture. The modified TAP (MTAP) behaves exactly as a classic TAP, except when it is an expansion/decompression process. The method has 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.

Independent and modular test architecture generators have been developed in the C language. Depending on the SoC features and constraints, the user can generate the appropriate TAM architecture in synthesizable format. These pieces of software make available a tool-box environment for automatic design of plug-and-play components, thus easing the design of SoC test architectures based on the CAS-BUS TAM and its developed instances. This kind of architecture is compliant with existing or under development IEEE standards, independently from any proprietary architecture.

Acknowledgments

Authors would like to thank Le Yin Keu for the development of the wrapper generator, in the framework of his DEA project, and Zakaryae Bekkouri and Gilles Richard for the development of the compression-decompression/expansion software, in the framework of their final undergraduate study project.

References

[1] Y. Zorian. Testing the monster chip. *IEEE Spectrum*, pages 54–60, July 1999.

[2] IEEE P1500 Standard for Embedded Core Test web site. http://grouper.ieee.org/groups/1500. Public informations.

[3] 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)*, Atlantic City (NJ), USA, September 1999.

[4] C. Feige and C. Wouters. Integration of structural test methods into an architecture specific core-test approach. In *2nd IEEE International Workshop on Testing Embedded Core-Based System-Chips (TECS)*, pages 5.2.1–5.2.8, Washington (DC), USA, October 1998.

[5] E. J. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousberg, and C. Wouters. A structured and scalable mechanism for test access to embedded reusable cores. In *IEEE International Test Conference (ITC)*, Washington (DC), USA, October 1998.

[6] P. Varma and S. Bhatia. A structured test reuse methodology for core-based system chips. In *IEEE International Test Conference (ITC)*, Washington (DC), USA, October 1998.

[7] S. K. Goel and E. J. Marinissen. TAM architectures and their implication on test application time. In *IEEE International Workshop on Testing Embedded Core-Based Systems (TECS)*, pages 3.3.1–3.3.10, Marina del Rey (CA), USA, May 2001.

[8] E. J. Marinissen and Maurice Lousberg. The role of test protocol in testing embedded core-based system ICs. In *IEEE European Test Workshop (ETW)*, pages 70–75, Constance, Germany, May 1999.

[9] K. Chakrabarty. Optimal test access architectures for system on a chip. *ACM Trans. Design Automation of Electronic Systems*, vol. 6, No (1): pp 26–49, January 2001.

[10] K. Chakrabarty. Design of system on a chip test access architectures under place-and-route and power constraints. In *IEEE/ACM Design Automation Conference (DAC)*, pages 432–437, Los Angeles (CA), USA, June 2000.

[11] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. Test wrapper and test acces mechanism co-optimization

for system on a chip. In *IEEE International Test Conference (ITC)*, pages 1023–1032, Baltimore, MD, USA, October 2001.

[12] E. Larsson and Z. Peng. An integrated system on a chip test framework. In *IEEE Design Automation and Test in Europe Conference (DATE)*, pages 138–144, Munich, Germany, March 2001.

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

[14] M. Benabdenbi, W. Maroufi, and M. Marzouki. Casbus: 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.

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

[16] B. Dervisoglu and J. Swamy. A novel approach for designing a hierarchical test access controller for embedded core designs in an SoC environment. In *4th IEEE International Workshop on Testing Embedded Core-Based System-Chips (TECS)*, pages 1.4.1–1.4.7, Montreal (QC), Canada, May 2000.

[17] 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), USA, November 1997.

[18] L. Whetsel and M.Ricchetti. Tapping into IEEE P1500 domains. In *5th IEEE International Workshop on Testing Embedded Core-Based Systems (TECS)*, Los Angeles (CA), USA, May 2001.

[19] M. Benabdenbi, W. Maroufi, and M. Marzouki. Testing taped cores and wrapped cores with the same test access mechanism. In *IEEE Design Automation and Test in Europe (DATE)*, pages 150–155, Munich, Germany, March 2001.

[20] W. Maroufi, M. Benabdenbi, and M. Marzouki. Solving the I/O bandwith problem in system on a chip testing. In *XIII Symposium on Integrated Circuits and System Design (SBCCI)*, pages 9–14, Manaus (AM), Brazil, September 2000.

[21] D. A. Lelewer and D. S. Hirschberg. Data compression. *ACM Computing Surveys (CSUR)*, vol. 19, No (3): pp 261–296, September 1987.

[22] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, vol. 30, No (6): pp 520–540, June 1987.

[23] 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 (TECS)*, pages 4.5.1–4.5.6, Montreal (QC), Canada, May 2000.

[24] J. van Beers and H. van Herten. Test features of a core-based co-processor array for video applications. In *IEEE International Test Conference (ITC)*, pages 638–647, Atlantic City (NJ), USA, September 1999.

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

[26] A. Jas and N. A. Touba. Test vector decompression via cyclical scan chains and its application to testing core-based designs. In *IEEE International test conference (ITC)*, pages 458–464, Washington (DC), USA, 1998.

[27] Synopsys. Synopsys design compiler family. http://www.synopsys.com/products/logic/logic.html. Synopsys, Inc.

[28] LIP6-ASIM. Alliance CAD system. http://www-asim.lip6.fr/alliance/. University Paris 6 - France.

[29] F. Wajsbürt, J.L. Desbarbieux, C. Spasevski, S. Penain, and A. Greiner. An integrated PCI component for IEEE 1355. In *European Multimedia Microprocessor Systems and Electronic Commerce Conference and Exhibition*, Florence, Italy, November 1997.

Mounir Benabdenbi was born in 1971 in Meknes, Morocco. He received the Masters Degree in Electronics and the D.E.A Degree in Microelectronics from the University of Montpellier. He is currently working towards his Ph.D. degree at LIP6 Laboratory under the direction of Meryem Marzouki. His research interests include Boundary Scan and SoC Testing.

Walid Maroufi was born in 1970 in Monastir, Tunisia. He received his Engineering degree from the Faculty of Electrical

Engineering of Monastir, Tunisia and his Ph.D. degree in Computer Science from the University Pierre et Marie Curie in Paris, France, where he spent afterwards one year of post-doctoral research. He is currently with Nortel Networks, Ottawa, Canada.

Meryem Marzouki was born in 1961 in Tunis, Tunisia. She received the D.E.A Degree and the Ph.D. Degree, both in Computer Science, from the National Polytechnical Institute of Grenoble in 1987 and 1991 respectively. Before that, she received the Engineering Degree in Computer Science too, from the University of Tunis, in 1986.

From 1987 to 1992, she was mainly involved in prototype validation of ICs in the framework of electron-beam testing. Since 1992, she has been a researcher with the CNRS (French National Research Center), in charge of the *Diagnosis of Complex System* Group at TIMA Laboratory in Grenoble, France, with artificial intelligence approaches for test and diagnosis as main research interests, until 1997, when she joined the LIP6 Laboratory in Paris. Since then, her research interests are related to high-level synthesis for testability and system on a chip testing.