CAS-BUS: A Scalable and Reconfigurable Test Access Mechanism for Systems on a Chip*

Mounir Benabdenebi Walid Maroufi
Meryem Marzouki
LIP6 Laboratory
Couloir 55-65, 4 Place Jussieu
75252 Paris Cedex France
Tel. (+33)1 44 27 39 67 - Fax. (+33)1 44 27 72 80
E-mail : {Mounir.Benabdenbi,Walid.Maroufi,Meryem.Marzouki}@lip6.fr

Abstract

This paper describes CAS-BUS, a P1500 compatible Test Access Mechanism for Systems on a Chip. The TAM architecture is made up of a Core Access Switch (CAS) and a test bus. The TAM characteristics are its flexibility, scalability and reconfigurability. A CAS generator has been developed, and some results are provided in the paper.

1 Introduction

The design of highly complex systems on single chips has been the new challenge faced by the design community for some years. Driven in one hand by the technological need for high-speed and large bandwidth applications and in the other hand by the commercial pressure for short time-to-market, the demand for systems-on-a-chip (SoCs) can be met thanks to the spectacular progresses in chip integration capacities, together with the availability of existing, highly specialized, hardware and software cores. These libraries of IP cores, either available in-house or commercialized on the market, are reused by embedding them into a single system chip, conceptually in the same way as integrated circuits were connected on PCBs.

However, while system geometry shrinking and design reusing allow impressive gains, SoC testing faces new set of problems, among which we can identify:

- testing cores with different functionnalities, coming from different companies,

- accessing cores from the system primary inputs and outputs,

- controlling whole SoC test process.

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

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 [1] have resulted in the P1500 standard proposal of a core test architecture, which main elements are, in its current development status:

- **test sources** to generate the cores 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** [2] 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. TAM architectures can be based on the use of the system bus [3] or on a specific test bus [4], [5].

In this paper we propose a P1500 compatible TAM architecture, named CAS-BUS, which falls into the second category. It is highly flexible, scalable and dynamically reconfigurable. The CAS-BUS can be connected to either internal and external sources and sinks (BIST or off-chip TPG). In addition, it supports hierarchical core architectures, where a core can itself embed other cores. More details on SoC test can be found in [6], [7] and [8].

2 The CAS-BUS architecture

The TAM architecture we propose (figure 1) is composed by two main elements:

- A **Core Access Switch (CAS)** which is a simple controller connected to each testable core through the P1500

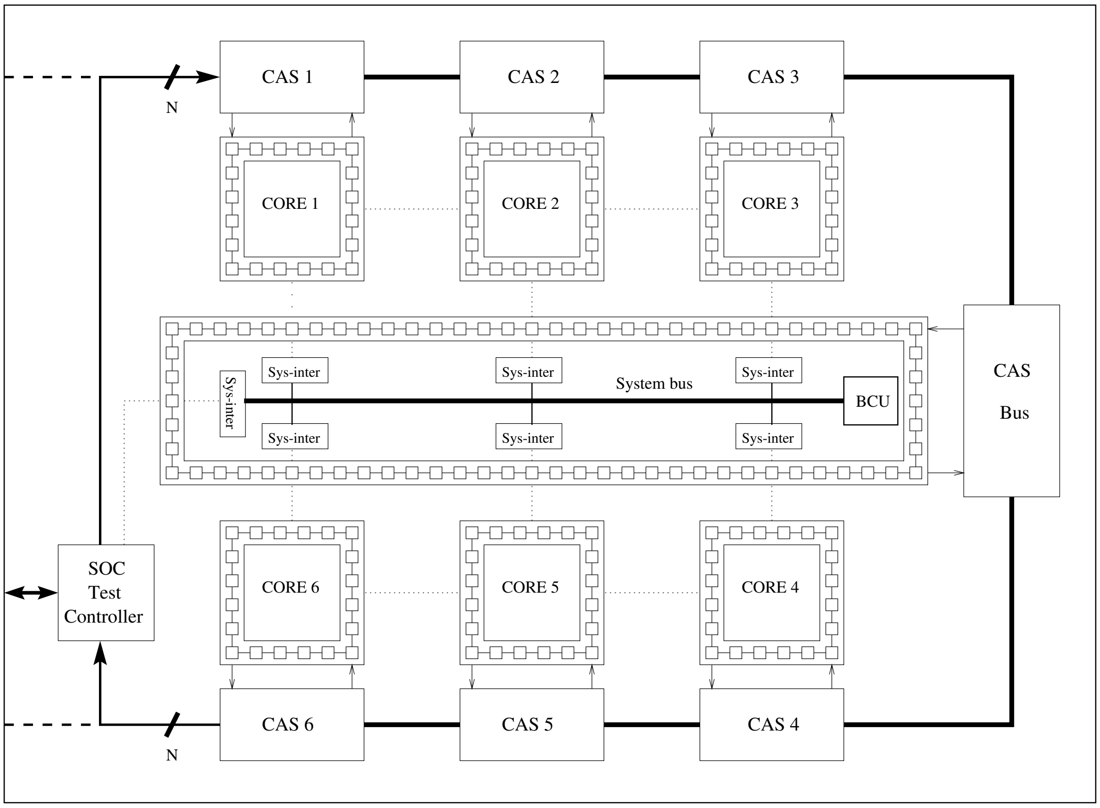

Figure 1. The CAS-BUS architecture

wrapper at its test terminals,

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

When the system bus is wrapped itself by a P1500 wrapper, it also has its dedicated CAS.

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 2 (a)),

- For BISTed cores, P is generally equal to 1 (figure 2 (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 2 (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 2 (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.

3 CAS architecture

The CAS is a simple configurable switcher which connects P wires out of the N test bus wires to the core test input terminals (e.g. scan-in inputs) and collects P wires from the core test output terminals (e.g. scan-out outputs), in order to connect them to the test bus. The connection to the test bus is made through the e_i inputs and s_i outputs, while the connection to the target core is made through the o_i and i_i tri-stated outputs and inputs. Figure 3 details the CAS architecture, where we can identify:

- An **instruction register**, used to initialize the adopted switching scheme. The width of this register obviously depends on N and P values. When all the instruction register bits are 0, the CAS is in a BYPASS mode, thus no test bus wire is connected to the core. The instruction registers of all the CASes are connected to each other through the first serial test bus wire (e0/s0) during the initialization phase,

- An **update mechanism** to configure the N/P switcher,

- The **N/P configurable switcher**. In configuration phase, the tri-stated switcher outputs and inputs are

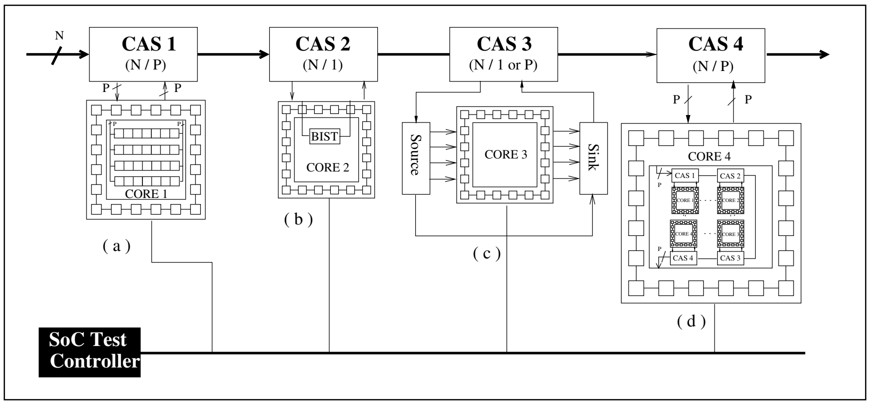

Figure 2. Test types supported by the CAS-BUS

switched to high impedance.

3.1 CAS functionl modes

The CAS has three functional modes:

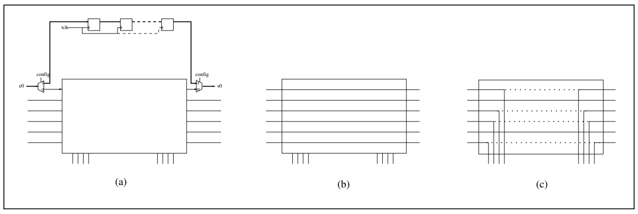

Figure 4. CAS modes

- The **CONFIGURATION** mode in which the instruction register is initialized (all c_i are at 1) (figure 4 (a)). The system test engineer may configure the wrapper independently, or may want to implement a tri-state mechanism, which allows to configure at the same time the CAS and the wrapper, by serially connecting the CAS and wrapper instruction registers, while synchronizing control signals of both elements through the SoC test controller (figure 3). The implementation of this mechanism is optional, but when integrated, it simplifies the overall SoC test architecture configuration during the testing process.

- The **BYPASS** mode, where all the test bus wires directly go through the CAS from e_i inputs towards s_i outputs (figure 4(b)) (instruction: 000...0),

- The **TEST** mode in which the N/P switcher is configured and the core is considered under test. Each instruction corresponds to a specific switch scheme. When P wires are switched to the core, the N-P remaining wires bypass the CAS (figure 4 (c)).

3.2 CAS Generation

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). A trade-off should be made on the value of N: the larger is the width of the test bus (N), the shorter is the overall test time. In addition, the number of possible combinations between N and P has a direct impact on the width of the CAS instruction register.

In order to reduce the number of combinations and thus the width of the instruction register, the following heuristic has been defined: *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 (figure 4 (c)) obviously limits the width of the test bus path between a core and its CAS. Thanks to this heuristic, we can completely construct a test bus wire path from an e input to an s output with the same control word $(c_0...c_k)$.

Some other heuristics are used to limit the total number *m* of combinations, which is equal to the number of required control instructions for the CAS.

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

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

3.3 Implementation and Results

A CAS architecture generator has been developed. It takes as parameters the N and P values, and provides a VHDL description of the CAS, which can be synthesized with a commercial synthesis tool. This generator is written in C, however, we have considered an alternative way of generation, which consists in describing a CAS architecture in generic VHDL.

In our experiments, we have synthesized various configurations of CASes using the Synopsys Design Analyzer

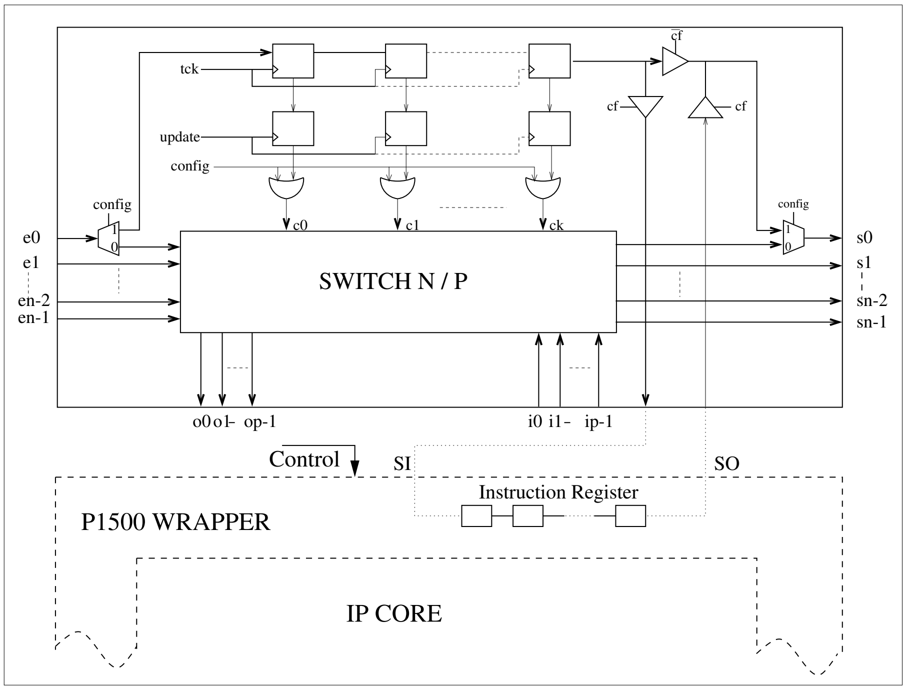

Figure 3. CAS architecture

tool. The resulting CAS architecture area is too small (table 1) compared to the SoC total area (millions of transistors) to influence the overall SoC test overhead. In fact, the major part of the test overhead will mainly be caused by the core internal DFT functionalities (including the P1500 wrapper). However, when the width of the test bus becomes important, the induced CAS-BUS overhead can be significant. A good trade-off between test time, test requirements and CAS-BUS overhead allows to choose an optimal width for the test bus.

N	P	m	k	# of gates.
3	1	5	3	16
4	1	6	3	23
4	2	14	4	64
4	3	26	5	118
5	1	7	3	28
5	2	22	5	85
5	3	62	6	205
6	1	8	3	33
6	2	32	5	134
6	3	122	7	280
6	5	722	10	1154
8	4	1682	11	4400

Table 1. CAS synthesis results

It is important to note that the width of the CAS instruction register, even when it is large, does not affect the test time, since the SOC test architecture configuration will only occur once at the beginning of a SoC testing session.

Moreover, two other ways to generate CASes are now under study. The first one consists in generating a highly optimized gate level description. The second one, which is much more optimized, considers a hardware architecture based on the use of pass transistors. These architectures are not detailed in this paper, but first experiments have shown that they solve the CAS area problem for large width test busses, even without restricting heuristics.

4 CAS-BUS Benefits

The CAS-BUS has multiple advantages due to its flexibility, scalability, reconfigurability and dynamic aspect:

- Supporting different core test methods (Scan, BIST, parallel test),

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

- Thanks to the CAS reconfigurability, the CAS-BUS architecture can be easily modified, 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,

- In case of maintenance test, it is possible to test some embedded cores while others are in normal functionning mode. This is very useful when, e.g., an embedded memory test is periodically required.

Compared to other approaches described in the literature, the CAS-BUS has the advantage of being independent of any industrial proprietary SoC architecture. Moreover, 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 [4], where the TAM and the wrapper are closely merged, leaving few freedom of decision to the system integrator. On the contrary, the CAS-BUS eases the SoC test architecture design by using plug-and-play CAS modules.

5 Conclusion

A P1500 compatible reconfigurable TAM architecture has been presented in this paper. It is characterized by its flexibility, scalability, reconfigurability, and dynamic aspect. The Core Access Switchers (CASes) and the test bus are the main components of the proposed architecture. Taking into account the test functionalities within the embedded cores and the required test performances, the test designer and the test programmer can make a decision concerning the test bus width, which then allows the generation of the reconfigurable CAS switchers.

Associated with a SoC central test controller, in charge of test data and control synchronization, and with the P1500 wrappers, the proposed CAS-BUS can offer a complete test architecture for the SoC.

Different TAM architectures can be addressed, in sequentiel order, within the same test program, in order to optimize test performances (time, pattern length, etc.). This represents the main advantage of the proposed reconfigurable CAS-BUS architecture.

References

[1] IEEE Computer Socity. IEEE P1500 Standard for Embedded Core Test. Technical report, IEEE, http://grouper.ieee.org/groups/1500, 1998.

[2] E. J. Marinissen, Y. Zorian, R. Kapur, T. Taylor, and l. Whetsel. Towards a standard for embedded core test: An example. In *International Test Conference*, Atlantic city, NJ, September 1999.

[3] P. Harrod. Testing reusable ip - a case study. In *International Test Conference*, pages 493–498, Atlantic city, NJ, September 1999.

[4] 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.

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

[6] R. K. Gupta and Y. Zorian. Introducing core-based system design. *IEEE Design and Test of Computers*, pages 15–25, October 1997.

[7] Y. Zorian, E. J. Marinissen, and S. Dey. Testing embedded-core based system chips. In *International Test Conference*, Washington D.C., USA, 1998.

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