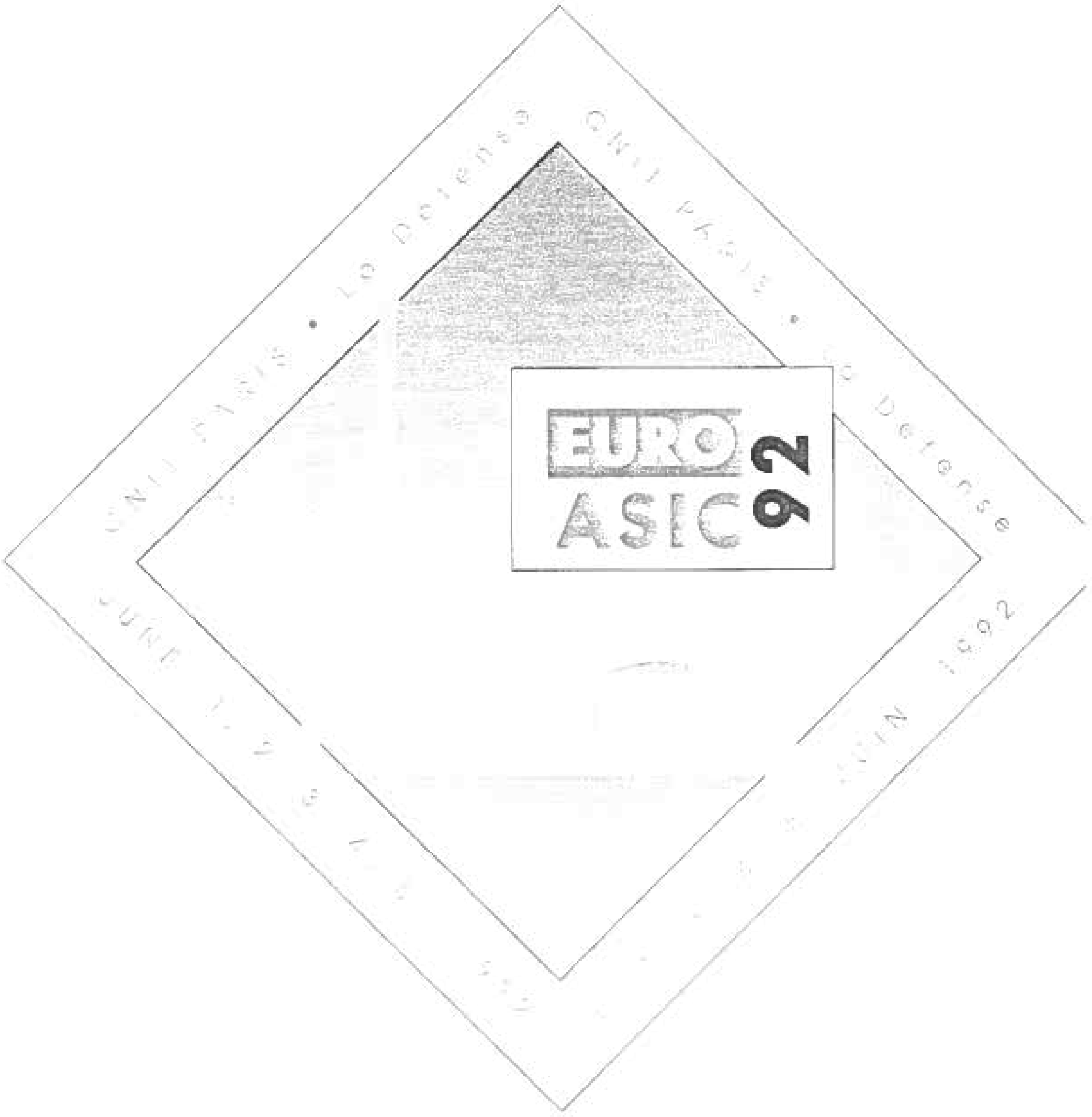
EURO ASIC '92



IEEE Computer Society Press

The Institute of Electrical and Electronics Engineers, Inc.

A MODULAR ARCHITECTURE FOR BIST OF BOUNDARY SCAN BOARDS

José M. M. Ferreira[1,2], Filipe S. Pinto[2], José S. Matos[1,2]

[1] FEUP / INESC
Rua dos Bragas
4099 Porto Codex - PORTUGAL

[2] INESC
Largo Mompilher, 22
4000 Porto - PORTUGAL

Abstract

A board-level BIST architecture for boards loaded with ASICs and VLSI components, compliant with the IEEE 1149.1 BST standard, is described. This BIST architecture consists of a test processor core, with an optimized architecture for controlling the board-level BST infrastructure, an optional system-level testability bus interface, to be included when a system-level test strategy is to be implemented, and a ROM containing the test program, which is automatically generated by an ATPG tool.

1 INTRODUCTION

This paper describes a modular architecture for integration of built-in self-test (BIST) resources in boards with BST. The need for built-in test (BIT) resources is first discussed, and it is shown that the Boundary Scan Test (BST) technology is able to satisfy the requirements identified for these resources. This test technology is already adopted by many commercial ASIC foundries, and PLD suppliers, which means that it has achieved the critical mass which will enable it to become *the* fundamental ASIC DFT framework for the 90's. The transition from BIT to BIST is then introduced, starting with the definition of an architecture for a dedicated test processor. Enhancement of this architecture to support a system-level test strategy is then discussed, followed by the presentation of an automatic test program generation (ATPG) tool for the described test processor. This BIST architecture is of particular interest to boards loaded with medium-to-high complexity ASICs and VLSI components.

1.1 THE NEED FOR BUILT-IN TEST (BIT) RESOURCES

The increasing complexity of testing has reached a point where traditional test techniques (in-circuit, functional) may simply become unusable, unless special BIT resources are available. While **miniaturization** seriously restricts the usefulness of in-circuit test techniques (main difficulty: physical access to internal nodes), the advances in **integration scale**, and the **widespread access to ASIC technology**, enable the design of very high complexity components, which in turn restrict the usefulness of functional test techniques (main difficulty: test program generation). The end result is that a design may easily become untestable.

The overall need for testability improvement has led to the development of the Boundary Scan Test (BST) standard [1], which effectively provides an answer to the need for BIT resources. Every BST component has a special cell associated with each functional pin, which allows complete controllability and observability of the corresponding electrical node. This set of cells (the boundary scan register) is controlled by an internal test architecture which interfaces the outside world by a four pin Test Access Port (TAP), and may be chained with the boundary scan registers of the remaining components to provide a board-level serial-access BIT infrastructure.

BST is essentially a board-level test technology, providing powerful BIT resources for interconnect testing. However, when combined with IC-level BIST functions, a high fault coverage structural test of the complete board becomes possible. The BST infrastructure is nearly equivalent to an electronic bed-of-nails, but without the drawbacks associated with in-circuit test techniques (no backdriving, no need for physical access). On the other hand, it solves part of the problem associated with functional test techniques (test program generation) by providing a gateway to BIST functions in complex components.

Integration of BST in an ASIC design flow is easy to accomplish, both because of its simplicity, and also because tools are becoming available to automate the inclusion of this test technology [2], [3]. These tools accept an input describing the required characteristics for the BST infrastructure, and automatically add the fundamental testability blocks. The resulting board-level BIT resources will be particularly important for those boards loaded with medium-to-high complexity ASICs and VLSI components implementing this test technology, where non-BST clusters will be of reduced size and complexity, or even non-existent.

1.2 ONE STEP FURTHER: BOARD-LEVEL BIST RESOURCES

Board-level BIT resources will therefore become available in an increasing number of designs, either resulting from the integration of BST in ASICs, or simply because a growing number of components will incorporate BST [4], [5], [6]. The question then becomes: What to do with these BIT resources?

BST will be combined with other test techniques in several scenarios: prototype validation / verification, production test, and field maintenance operations. Enhancement of functional test techniques is also possible [7], [8]. But the next logical step is to include the blocks which will enable the use of these BIT resources for board-level BIST functions. The need for board-level BIST is being recognized for a growing number of designs [9], [10], [11], and the widespread use of the standard BST testability infrastructure will undoubtedly reinforce this tendency.

2 A MODULAR ARCHITECTURE FOR BOARD-LEVEL BIST RESOURCES

The board-level BIST resources described in this paper consist of a dedicated test processor, which executes a test program stored in ROM (internal or external). Additionally, and when the board is to be integrated in a system, an optional system-level testability bus interface may also be present.

An automatic test program generation (ATPG) tool provides an output file where all the elementary test operations are specified, generated from an input describing the board netlist, the BST infrastructure in each component, and the clusters of non-BST components present.

The description of the BST infrastructure in each component must provide all the information required both for board and component-level test procedures. Component test will take place either through BIST functions, or by successively applying a specified set of test vectors.

Clusters of non-BST components are tested according to a technique known as *virtual cluster testing* [12]. The complete set of test vectors for these clusters must be generated externally, and should be specified in the input description file of each cluster.

The test program output by the ATPG tool is specified in terms of a low-level language (set of commands), able to represent all the elementary operations required to control the BIT resources available. The complete set of low-level commands is described in table 1 (N represents the value loaded in an internal counter), and constitutes the instruction set of a dedicated test processor core.

Procedure	Instruction
Control of the BST infrastructure	
Applies N test clock cycles.	NTCK
N bits will be shifted into the BST chain. Bits shifted out of the BST chain are not compared.	NSHF
N bits will be shifted into the BST chain. Bits shifted out of the BST chain are compared with their expected value. A mask is used to discard don´t care bits.	NSHFCP
Forces an asynchronous reset through the active /TRST output.	TRST
Forces a state transition in the internal BST logic of each component.	TMS0, TMS1
Selects which TAP will be controlled by the following instructions.	SELTAP0, SELTAP1
Control of internal processor resources	
Loads an internal counter with the number of test clock cycles to be applied.	LD CNT, N
Selects the active error flag.	SERFLG0,... ..., SERFLG7
Leaves the normal test program flow, based on the state of the active error flag.	JPE Address, JPNE Address
Terminates the execution of a test program.	HALT
Test execution synchronization	
Forces a logical value (0,1) on the specified synchronism output (A,B).	SSA0, SSA1, SSB0, SSB1
Waits for a logical value (0,1) on the specified synchronism input (A,B).	WSA0, WSA1, WSB0, WSB1

Table 1: Elementary operations required for the test processor core.

2.1 THE TEST PROCESSOR CORE

The identification of the required instruction set led to the definition of a dedicated architecture for a test processor core, shown in figure 1.

This architecture is able to implement the complete instruction set identified in table 1, and allows an optimized execution of test programs for boards with one or two BST chains. Synchronization with external equipment is supported, for those cases where the BIT resources are used in conjunction with external test resources.

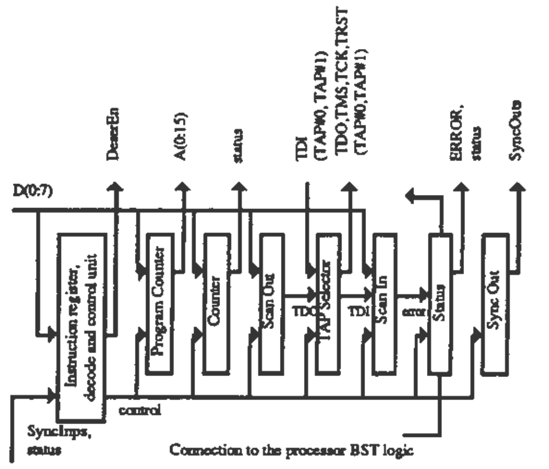


Fig.1: Architecture of a dedicated test processor core, for boards with BST.

Execution of the test program will provide a simple pass / fail result, available through an output pin. However, an internal status register provides two groups of 8 error flags (one for each board BST chain), which may be used to identify the fault(s) detected. The contents of this register are accessible to the system-level testability bus interface, which may then send this information to a higher level test processor.

2.2 THE SYSTEM-LEVEL TESTABILITY BUS INTERFACE

Definition of a system-level testability bus is still under way, with the Module Test and Maintenance (MTM) Bus being expected as a standard in the near future [13]. Other alternatives are also possible [9], [11], [14], one of them simply consisting of extending the BST bus to system-level. The main disadvantage of this last alternative consists of the long scan chains which would result from serially connecting the BST chain in each board. This disadvantage will however disappear if each board has its own *test processor with BST*, in which case the system test processor will only control the BST infrastructure of each board test processor.

A system-level testability bus interface was therefore implemented simply by adding a BST infrastructure to the test processor core, and including the additional interaction blocks required. The system-level BST chain will in this case contain a number of components equal to the number of boards present (one test processor in each board). One of the commands supported by the BST infrastructure in each test processor is *Run Board Test* (besides the mandatory *Extest*, *Bypass*, and *Sample / Preload*), which enables the system-level test processor to order the execution of BIST on the selected boards. Notice that this approach has the additional advantage of using only one testability standard both at board and system levels. As a result, *the same test processor* may be used at different hierarchical levels, as illustrated in figure 2.

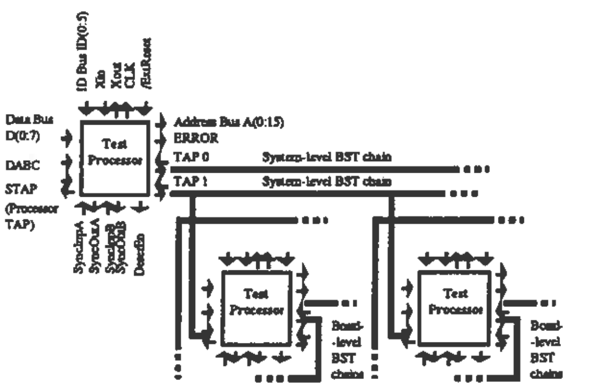
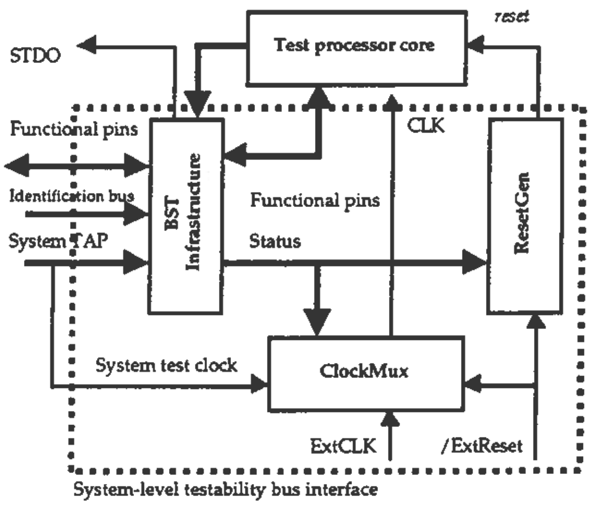
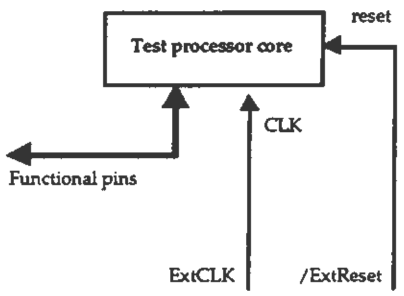


Fig.2: Hierarchical test strategy, using the same test processor at different levels.

Initialization of the test processor core, and test program execution, may have its origin locally (on the board), or remotely (by the system test processor). Two additional interaction blocks are therefore required, which multiplex the initialization signal (local reset, or through the BST infrastructure), and the clock source (local clock, or system test clock), as illustrated in figure 3.



(a) BIST resources for a system-level test strategy.



(b) Stand-alone BIST resources.

Fig.3: Test processor, and optional system-level testability bus interface block.

Additionally, the system-level testability bus interface includes a 6 pin identification bus, which allows each board to be assigned a specific address. This identification bus allows the system test processor to check for proper board placement.

2.3 AUTOMATIC TEST PROGRAM GENERATION (ATPG)

The proposed board-level BIST architecture will only be feasible if the test processor program is automatically generated. When an internal ROM is to be used for test program storage, the corresponding block should also be completely defined by the ATPG tool. In this case, a one-chip solution for board-level BIST of BST boards becomes possible [15].

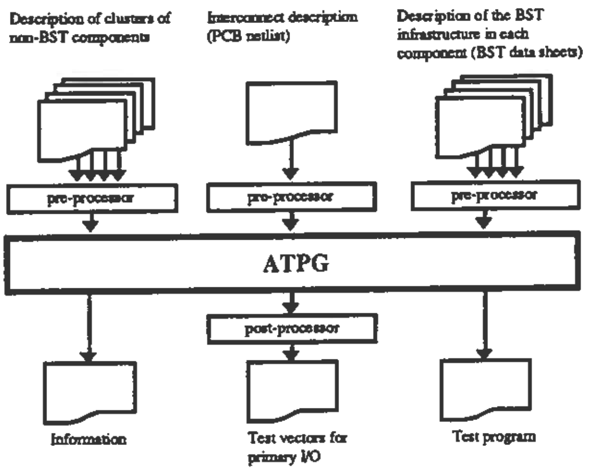

Description of clusters of non-BST components
Interconnect description (PCB netlist)
Description of the BST infrastructure in each component (BST data sheets)
pre-processor
pre-processor
pre-processor
ATPG
post-processor
Information
Test vectors for primary I/O
Test program

Fig.4: Data flow diagram for the ATPG tool.

The ATPG tool requires an input information describing the board netlist, the BST infrastructure in each component, and the clusters of non-BST components present (including the corresponding test vectors). The test program for the dedicated test processor is then generated, together with the set of test vectors required for primary I/O pins. The data flow diagram for the ATPG tool is illustrated in figure 4.

Preprocessors are under development to cope with advances in the standardization of data representation formats, specially for the description of the BST implementation in each component [16], [17].

The ATPG tool uses a fault model consisting of open and short-circuit faults, and assumes that the values captured by the test infrastructure, under faulty conditions, are not known. This fact will essentially affect the diagnostic resolution available (which is itself one of the limitations of the BST technology), but will not impair the fault detection capability of the resulting test program (the main requirement for a pass / fail test).

ATPG proceeds in three main steps, consisting of test program generation for the board-level BST infrastructure, for interconnect testing, and for component testing.

Testing the BST infrastructure consists essentially of shifting through the instruction register in each BST component [18], which will toggle each TDO-TDI interconnect.

Interconnect testing will take place first for full BST interconnects, and then for cluster interconnects. Open faults in full BST interconnects are tested by successively forcing a **0**, and a **1**, from each driving pin in every interconnect. Test pattern generation for detecting short faults among full BST interconnects is done according to an algorithm similar to the self diagnosis algorithm [19]. A slight modification is included, in order to guarantee that each vector will always keep half the total number of interconnects at **0**, and the other half at **1** [20]. Cluster interconnects are tested according to a set of test vectors generated externally, which must be specified in the respective description file.

Component testing will then take place, first for those BST components with BIST capability, and then for the remaining BST components.

The test processor program output by the ATPG tool is completely specified in terms of the elementary operations described in table 1.

3 CONCLUSION

A modular architecture for board-level BIST resources was described, and has been implemented. These resources consist of a dedicated test processor core (optimized for controlling the board-level BST infrastructure), a system-level testability bus interface, and a ROM containing the test program. Test program generation is supported by an ATPG tool, which produces an output code based on the instruction set of the dedicated test processor core.

The described architecture has been manufactured on a 1,5 µ CMOS technology. Overhead for the system-level testability bus interface will be minimized if a library of BST building blocks is available. Main characteristics of the manufactured prototype are: 68-pin LCC package (50-cell boundary scan register, 6-bit identification bus, 16-bit address bus, 8-bit data bus), and a 25 MHz clock frequency.

The described architecture is of particular importance for boards loaded with medium-to-high complexity ASICs and VLSI components compliant with the IEEE 1149.1 BST standard, where the resulting BIT resources will provide high fault coverage, with minimum test program length.

REFERENCES

[1] IEEE Standards Board, *IEEE Std 1149.1: Standard Test Access Port and Boundary Scan Architecture*, May 1990.

[2] M. Muris, "Integrating Boundary Scan Test Into an ASIC Design Flow," in *Proc. of the IEEE International Test Conference*, 1990, pp. 472-477.

[3] D. Chiles and J. DeJaco, "Using Boundary Scan Description Language in Design," in *Proc. of the IEEE International Test Conference*, 1991, pp. 865-868.

[4] W. Bruce, M. Gallup, G. Giles, and T. Munns, "Implementing 1149.1 on CMOS Microprocessors," in *Proc. of the IEEE International Test Conference*, 1991, pp. 879-886.

[5] L. Whetsel, "An IEEE 1149.1 Based Logic / Signature Analyzer in a Chip," in *Proc. of the IEEE International Test Conference*, 1991, pp. 869-878.

[6] R. G. Bennetts and A. Osseyran, "IEEE Standard 1149.1-1990 on Boundary-Scan: History, Literature Survey, and Current Status," *Journal of Electronic Testing: Theory and Applications*, Vol.2, Nº 1, pp. 11-25, March 1991.

[7] M. Lefebvre, "Functional Test and Diagnosis: A Proposed JTAG Sample Mode Scan Tester," in *Proc. of the IEEE International Test Conference*, 1990, pp. 294-303.

[8] K. Wagner and T. Williams, "Enhancing Board Functional Self-Test by Concurrent Sampling," in *Proc. of the IEEE International Test Conference*, 1991, pp. 633-640.

[9] J. C. Lien and M. A. Breuer, "A Universal Test and Maintenance Controller for Modules and Boards," *IEEE Transactions on Industrial Electronics*, Vol. 36, Nº 2, pp. 231-240, May 1989.

[10] B. Dervisoglu, "Towards a Standard Approach for Controlling Board-Level Test Functions," in *Proc. of the IEEE International Test Conference*, 1990, pp. 582-590.

[11] N. Jarwala and C. Yau, "Achieving Board-Level BIST Using the Boundary-Scan Master," in *Proc. of the IEEE International Test Conference*, 1991, pp. 649-658.

[12] P. Hansen, "Assessing Fault Coverage in Virtual In-Circuit Testing of Partial Boundary-Scan Boards," in *Proc. of the European Test Conference*, 1991, pp. 393-396.

[13] Test Technology Technical Committee of the IEEE Computer Society, *IEEE P1149.5 Std: Standard Backplane Module Test and Maintenance (MTM) Bus Protocol*, Draft 0.9, March, 1991.

[14] D. Bhavsar, "An Architecture for Extending the IEEE Standard 1149.1 Test Access Port to System Backplanes," in *Proc. of the IEEE International Test Conference*, 1991, pp. 768-776.

[15] J. M. Ferreira, J. S. Matos and F. S. Pinto, "Automatic Generation of a Single-Chip Solution for Board-Level BIST of BST Boards," *Proceedings of the European Design Automation Conference (EDAC)*, March, 1992.

[16] K. Parker and S. Oresjo, "A Language for Describing Boundary-Scan Devices," *Journal of Electronic Testing: Theory and Applications*, Vol. 2, Nº 1, pp. 43-75, March 1991.

[17] C. Maunder, "Languages to Support Boundary-Scan Test," in *Proc. of the IEEE International Test Conference*, 1991, p. 1104 (*panel summary*).

[18] F. de Jong and F. van der Heyden, "Testing the Integrity of the Boundary Scan Test Infrastructure," in *Proc. of the IEEE International Test Conference*, 1991, pp. 106-112.

[19] W. Cheng, J. Lewandowski and E. Wu, "Diagnosis for Wiring Interconnects," in *Proc. of the IEEE International Test Conference*, 1990, pp. 565-571.

[20] J. Robinson and M. Cohn, "Counting Sequences," *IEEE Transactions on Computers*, Vol. C-30, Nº 1, January 1981, pp. 17-23.