ITC 92
pp. 217-223

A BOUNDARY SCAN TEST CONTROLLER FOR HIERARCHICAL BIST

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

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

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

Abstract

A test controller for BIST of Boundary Scan Boards is described. It consists of a test processor core, with an optimized architecture for controlling the board-level BST infrastructure, and a system level testability bus interface, allowing the implementation of a hierarchical test strategy. Automatic test pattern generation for this dedicated processor simplifies the task of providing a board-level BIST solution.

1 INTRODUCTION

The increasing complexity of testing has reached a point where traditional test techniques (in-circuit, functional) may simply become unusable, unless special resources for built-in test are available. While miniaturization seriously restricts the usefulness of in-circuit test techniques (lack of physical access to internal nodes), the advances in integration scale, and the widespread access to ASIC technology limit the usefulness of functional testing (complex test program generation). The end result is a growing difficulty in testing both current and future designs.

The overall need for testability improvement has lead to the development of the Boundary Scan Test (BST) standard [1], which effectively provides an answer to the need for built-in test resources. Every BST component has a special cell associated with each functional pin, allowing complete controllability and observability of the corresponding electrical node. This set of cells (the boundary scan register), and a test logic controller, interface to the outside world by a four pin Test Access Port (TAP). By chaining together the boundary scan registers of board components, a serial-access, board-level test infrastructure is available, which resembles an "electronic bed of nails".

Provision of a board-level Boundary Scan Test (BST) infrastructure is becoming a common feature for an increasing number of designs, either resulting from the automated inclusion of BST in ASICs [2], [3], or simply because a growing number of standard components will incorporate BST. This test technology is already adopted by many commercial ASIC foundries and PLD/FPGA suppliers, and is quickly achieving the critical mass needed to become a fundamental ASIC DFT framework for the 90's. There is a growing interest for board-level built-in self-test (BIST) functions using the BST infrastructure [4], [5], and a widespread use of this test technology will undoubtedly reinforce such interest.

This paper describes an architecture for providing built-in self-test (BIST) that can be extended from board to system level, making use of available BST resources. The architecture of a dedicated test processor capable of efficiently controlling the board level BST infrastructure is described. Enhancement of this architecture to support a system-level hierarchical test strategy is then discussed, followed by the presentation of an automatic test program generation (ATPG) tool for the described test processor.

2 BOARD-LEVEL BIST

Testing a board involves testing its components and their interconnects. Although especially suited for the latter, BST can be of assistance in other areas like the test of clusters of non-BST components. Furthermore, the ability to provide a gateway to BIST functions in complex components is of great importance. When combined with IC-level BIST functions, a high fault coverage structural test of the complete board becomes possible.

The powerful features of the BST infrastructure, and the fact that it is supported by an IEEE standard, have lead to a growing interest in using it as a vehicule for the development of BIST techniques for complex boards. These techniques rely on local control of the BST infrastucture, by means of special components like an on-board test processor, capable of executing its own test program [6], [7].

In this paper, the architecture of an on-board test processor is described. Its instruction set directly implements the elementary operations required to control the board-level BST infrastructure (two boundary scan chains are supported). This component is equipped with its own boundary scan register and TAP controller, and allows an hierarchical configuration where the same test processor may be used at different levels. The test program for this dedicated test processor, stored in internal

or external ROM, is automatically generated by a TPG tool that is also described.

3 PROCESSOR ARCHITECTURE

The processor architecture and instruction set were defined after the specification of a minimum, and yet complete, set of elementary operations, able to efficiently test the board through its BST infrastructure. Such set of elementary test operations required for the test controller is shown in Table 1. Also shown are the basic processor instructions that allow execution of those operations.

Although it is expected that the MTM (Module Test and Maintenance) bus will become the future standard [8], definition of a system-level testability bus standard is still under way. The architecture of the test controller distinguishes between the test processor core and the system level testability bus interface, therefore allowing for the same core to be used with a standard system test bus, when one is available. As will be explained in section 5, the test processor presently uses a Boundary Scan infrastructure to extend BST to the system level.

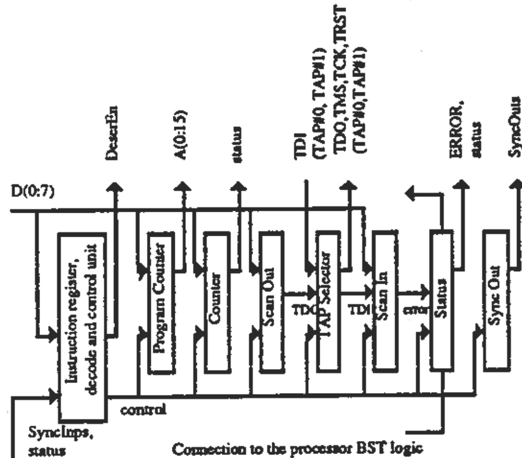


Fig.1: Architecture of a dedic ated test processor core, for boards with BST.

Figure 1 shows the architecture of the test processor core. 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 BST infrastructure is used in conjunction with external test resources.

Execution of the test program provides 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 types of detected fault(s). The contents of this register are accessible to the system-level testability bus interface, making this information available to a higher level test processor.

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 test operations required for the test controller.

The data bus width is 8-bit. Fewer memory access cycles would be required, if a larger data bus width was selected, but the efficiency of memory usage would then be lower. In fact, one word is required for storing each instruction opcode, leaving a large number of bits potentially unused. The execution speed is improved, because memory access cycles, and the control of the BST infrastructure, are performed concurrently. This solution allows that only nine clock cycles are needed for each 8 bits of data to be inserted in the BST chain, although three memory access cycles are performed (data, expected results, mask information).

The complete set of pins provided by the test processor

is shown in figure 2, grouped according to their functionality. Direct access to the board-level BST infrastructure is possible by controlling the logic level on the *DABC* pin. The *DeserEn* output will be high when the data being shifted through the board-level BST chain is compared to their pre-defined values. This pin may be used for enabling an external deserializer, which allows this test processor to be used as the central resource of an off-board test controller.

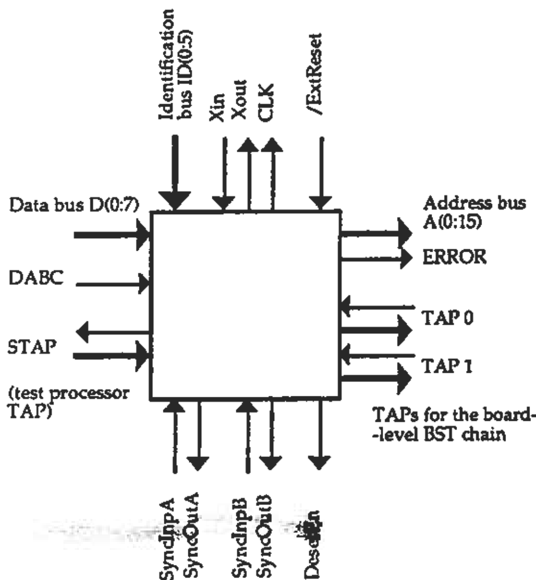


Fig.2: Pin functions of the test processor.

4 TEST PROGRAM GENERATION

An ATPG tool is of fundamental importance to establish the usefulness of a test processor, and its specification must be closely related to the definition of the processor architecture. The test controller model assumed for the ATPG process corresponds to the architecture defined in Section 3, and supports the set of elementary operations presented in table 1. It should be noted that the usefulness of the ATPG tool to be described is not restricted to the specific architecture that was presented above. In fact the same test controller model could be implemented by external test equipment, or any other processor capable of emulating this set of elementary operations.

The majority of the papers published in this domain [9-17] assume an interconnect fault model which includes stuck-at and short faults. Most authors assume that open and short faults exhibit a deterministic behaviour, whereby a floating input is usually assumed to capture a logic 1, and shorted outputs are assumed to behave in a wired-and or wired-or manner. Although this may be reasonable for specific cases, the fact is that real examples will seldomly exhibit a fixed-pattern behaviour. Experiences conducted with BiCMOS circuits with BST have shown that the behaviour under short fault conditions is essentially dependent on the number, and on the logic levels, of the shorted outputs (which is reasonable for outputs with the same source/sink strength). A diagnostic resolution limitation inherent to the BST technology compounds this problem, since a digital probe (the BS cell) is used to capture an analog value (the result of a short fault). A reasonable conclusion is that the assumption of a fixed-pattern behaviour under fault conditions will be of reduced practical interest.

The fault model assumed by our ATPG tool consists of open and short faults. It is assumed that a floating input will capture an unpredictable value, and that short faults will behave in an unpredictable manner. However, it is assumed that the values captured at the receiving nodes of at least one of the shorted interconnects will produce results which differ from those corresponding to the fault-free situation.

4.1 TEST PROTOCOL

The test protocol used by the ATPG tool follows the work described in [16], [18], [19] for testing boundary scan boards. It consists of three main steps, addressing the board-level BST infrastructure, the board interconnects, and the components.

The main testing actions for checking the BST infrastructure consist of shifting through the instruction registers, and through the (available) identification registers, according to a procedure described in [20]. When two BST chains exist, this test step is repeated independently for each BST chain in sequence.

Interconnect testing is divided into two main parts: full BST interconnect testing (for those interconnects in which every node is either a BST pin, or a primary I/O pin), and cluster interconnect testing, which will take place when clusters of non BST components are present. Full BST interconnects are tested for open and short faults.

Open faults in each interconnect are tested by successively forcing a 0, and a 1, from every driving pin in sequence, and checking the responses captured at every receiving node. All the interconnects are tested in parallel. Notice that stuck-at faults need not be explicitely considered, since any fault of this type will be detected at this step.

Short faults in full BST interconnects are tested by applying a sequence of logic values generated according to an algorithm which aims at achieving two main goals: an equal number of 0s and 1s should be applied to each interconnect by the set of test vectors generated, and each of these test vectors should guarantee that half of the interconnects has a 1 applied, while the other half has a 0. The first condition is related to the assumption of a non-fixed pattern behaviour (wired-and, wired-or) for short faults, and the second condition tries to maximize the probability of detecting a short fault to non-full-BST

interconnects. This algorithm proceeds by generating a sequence of codes similar to those generated by the self-diagnosis algorithm [15], but the Hamming distance between sucessive codes is a modified version of the one in [21].

Cluster interconnect testing is performed through the surrounding BST infrastructure, according to a technique described in [19] as virtual cluster testing. This step uses a set of test vectors generated externally for each cluster, and takes place following the test of full-BST interconnects.

Test action	One chain	Two chains
Shift the test vector into BST chain 0	•	•
Proceed to *Select-DR-scan*	•	•
Select BST chain 1		•
Shift the test vector into BST chain 1		•
Proceed to *Select-DR-scan*		•
Rise *BST_READY*	•	•
Wait for *ATE_READY* to rise	•	•
Proceed to *Shift-DR*	•	•
Select BST chain 0		•
Proceed to *Shift-DR*		•
Lower *BST_READY*	•	•
Wait for *ATE_READY* to be lowered	•	•

Table 2: Sequence of test actions for interconnect testing.

All test vectors generated for interconnect testing (either full-BST, or cluster interconnects) are applied with the test logic in the external test mode (*Extest* instruction), by cycling through the sequence of test actions described in table 2, where one or two BST chains are considered. Each interconnect test vector requires that a fixed sequence is followed for the capture/update operations in each BST chain. The *BST_ready* and *ATE_ready* signals are used to synchronize these operations with external test equipment used for the primary I/O pins (these signals are related to the SS and WS instructions described in the set of elementary test operations presented in table 1).

4.2 DATA FLOW

The data flow illustrated in figure 3 shows the three types of information required for the ATPG process: the description of the BST implementation in each component, the description of the board interconnects (board netlist), and the description of externally generated test vectors available for testing clusters of non-BST components.

Preprocessors are used to cope with advances in the standardization of data representation formats, specially for the description of the BST implementation in each component [22], [23]. The set of test vectors generated may be complemented with test vectors described in additional input files, which will be used when simple clusters of non-BST components, or BST components without BIST, are present.

4.3 ATPG OUTPUT

The ATPG tool generates the complete set of test vectors that will be used for testing a board. This set includes the serial test vectors for the scan chains, and the parallel test vectors for the primary I/O pins. The set of test vectors generated for the scan chains are converted to the corresponding test controller instruction formats (NSHF and NSHFCP), and are then interleaved with the additional elementary operations described in table 1, to produce the complete test processor program. These additional operations include the necessary synchronization sequences with the external test equipment used for the setting of primary I/O pins.

4.4 USING BOARD-LEVEL TPG RESOURCES

The developed ATPG tool produces a test program addressing all the steps required for testing a board with boundary scan, and assumes that the operating modes provided by the test logic in each component are those that correspond to the mandatory and optional instructions described in the 1149.1 standard.

Clusters of non-BST components are tested using the virtual cluster testing technique [20], using a set of test vectors generated by an external ATPG tool. Two main drawbacks may be identified in this approach:

- Virtual cluster testing is done by serialising each cluster test vector, which is applied through the BST infrastructure surrounding the cluster. Shifting in a new vector, while the responses to the previous one are shifted out, requires a total number of test clock cycles which is approximately $N \cdot M^{\dagger}$ (N test clock cycles for each test vector).

- Each serialized test vector must be resident in the memory containing the test program, seriously restricting the size and complexity of the clusters. A cluster requiring a set of 4.000 test vectors, inserted in a scan chain with only 100 cells, would require approximately 150 Kbytes of memory capacity.

† M is the number of test vectors, and N is the number of cells in the scan chain.

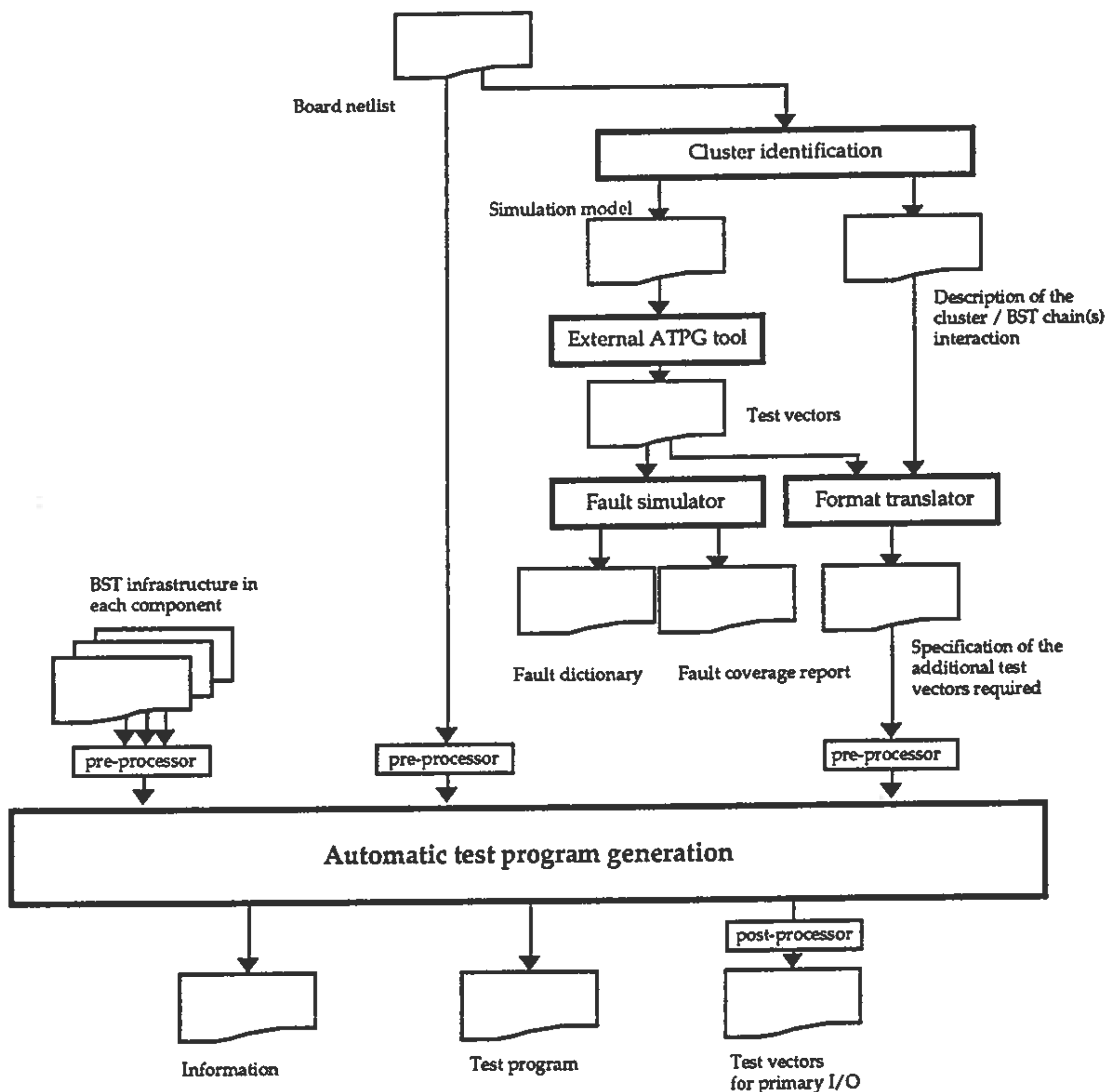


Fig.3: ATPG data flow.

The solution to these problems may be found in the use of embedded board-level TPG resources, allowing pseudo-random pattern generation (PRPG) and signature-analysis (SA). Besides being able to generate a new test vector for each test clock cycle, this alternative will not use the memory of the test processor for test vector storage. Components with a BST infrastructure providing additional operating modes for PRPG and SA are already available on the market, and more sophisticated components have been announced. Examples are the *Digital Bus Monitor* from Texas Instruments [24], which allows powerful on-board resources for PRPG, and response capture/compaction, and also a *Static RAM Tester* chip developed by SGS-Thomson [25], which allows local TPG for static RAM arrays according to a predefined algorithm.

While providing powerful board-level built-in TPG resources for clusters of non-BST logic, these new BST components also raise new challenges to an ATPG tool for boundary scan boards. The more powerful operating modes provided by these components do not require modifications to the test controller model considered, but enhancements will have to take place on the ATPG tool, and additional input information will be required.

Exploitation of the board-level built-in TPG resources means that no external ATPG tool for these clusters may be required, and a structured approach must be defined for specifying how to generate the test program sequence

(using the instruction set in table 1) that will trigger the local TPG functions for the clusters. An additional branch will have to be present on the data flow diagram shown in figure 3, providing the ATPG tool with input information for generating this test program sequence. Part of this additional information may be related to the functionality and characteristics of the cluster, and to its interaction with the remaining components on the board, eventually meaning that a substantial human intervention may have to take place. Work is being done on the characterization of the information required for generating the test program sequences for BST components providing board-level built-in TPG resources, and on the feasibility of an automated extraction of this information from the design data base.

5 HIERARCHICAL TEST

Extending BIST functions to the system level is implemented by adding special initialization and interaction functions to the test processor core, and using the Test Access Port of the test controller on-chip BST infrastructure as the system-level testability bus interface.

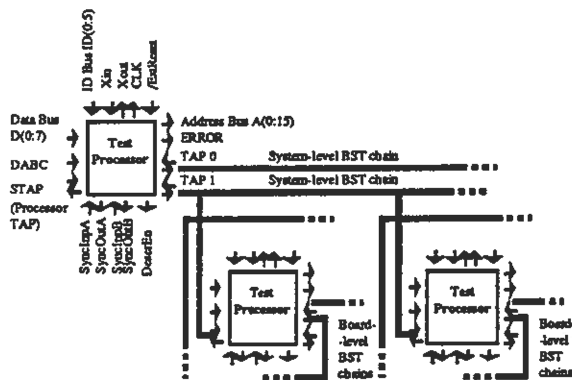
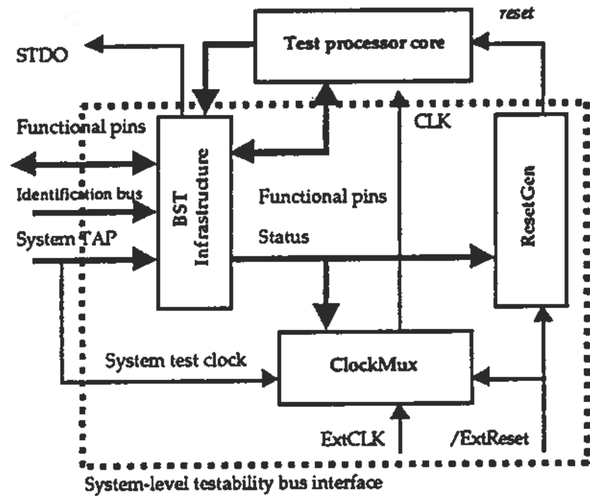


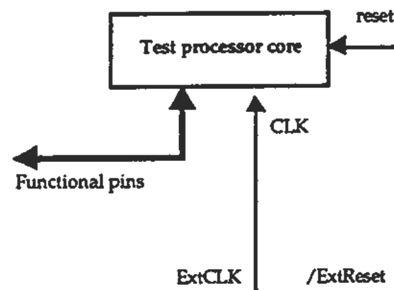
Fig.4: Hierarchical test strategy, using the same test processor at different levels.

A system-level testability bus interface may therefore be 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 will enable the system-level test processor to order the execution of BIST on the selected board. 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 4.

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 should therefore be present, which will multiplex the initialization signal (local reset, or through the BST infrastructure), and the clock source (local clock, or system test clock), as shown in figure 5.



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



(b) Stand-alone BIST resources.

Fig.5: 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.

6 CONCLUSION

An architecture for BIST on boards equipped with Boundary Scan was described. It includes a dedicated test processor core (with an optimized architecture for controlling the board-level BST infrastructure), a system-

level testability bus interface, and a ROM containing the test program. A prototype version of this architecture was implemented using a commercial ASIC design system (SOLO 1400) and was manufactured by ES2 (European Silicon Structures). The circuit, housed in a 68-pin LCC package, uses 1.5 μm CMOS technology and runs on a 25 MHz clock.

Relevant aspects of a TPG tool designed to automate the generation of test programs for the processor were presented. The test program produced by this tool makes use of the mandatory instructions described in the 1149.1 standard (*Extest, Bypass, Sample/Preload*), and is able to detect any open or short fault present on board-level interconnects. A test program segment devoted to component testing is also produced, for those components supporting the optional instructions described in the standard (*Intest, Runbist, Idcode, Usercode*).

The hierarchical extension for system-level test was discussed, and a solution was proposed, based on the use of the processor's BST Test Access Port as the system-level testability bus. An interesting feature of this approach is that one single testability standard can be used, allowing the same test processor to be used at different hierarchical levels.

7 ACKNOWLEDGEMENTS

This work has been partially supported by the *European Strategic Programme for Research and Development in Information Technology* (ESPRIT) under contract number 2478 (Research into Boundary Scan Test Implementation), and also by the *Junta Nacional de Investigação Científica e Tecnológica* (JNICT), under contract number PMCT/C/TIT/937/90. The manufacture of the circuit was funded by the EUROCHIP Program. The authors gratefully acknowledge many helpful discussions with people from the ESPRIT 2478 consortium, and in particular with Frans de Jong, from the ED&T group of the Philips Research Labs.

REFERENCES

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

[2] Muris, M., 1990. Integrating Boundary Scan Test Into an ASIC Design Flow. *IEEE ITC Proceedings*, 1990.

[3] Chiles, D. and DeJaco, J., 1991. Using Boundary Scan Description Language in Design. *IEEE ITC Proceedings*, 1991, pp. 865-868.

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

[5] Dervisoglu, B., 1990. Towards a Standard Approach for Controlling Board-Level Test Functions. *IEEE ITC Proceedings*, 1990, pp. 582-590.

[6] Jarwala, N. and Yau, C., 1991. Achieving Board-Level BIST Using the Boundary-Scan Master. *IEEE ITC Proceedings*, 1991, pp. 649-658.

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

[8] IEEE P1149.5 Std. 1992. *Standard Backplane Module Test and Maintenance (MTM) Bus Protocol*. Test Technology Technical Committee of the IEEE Computer Society, Draft 0.9.4, March, 1992.

[9] Goel, P. and McMahon, M. 1982. Electronic Chip-In-Place Test. *IEEE International Test Conference Proceedings*, 1982, pp. 83-90.

[10] Wagner, P. 1987. Interconnect Testing with Boundary Scan. *IEEE International Test Conference Proceedings*, 1987.

[11] Hassan, A., Rajski, J. and Agarwal, V. 1988. Testing and Diagnosis of Interconnects Using Boundary Scan Architecture. *IEEE International Test Conference Proceedings*, 1988, pp. 126-137.

[12] Hassan, A., Agarwal, V., Rajski, J. and Dostie, B. 1989. Testing of Glue Logic Interconnects Using Boundary Scan Architecture. *IEEE International Test Conference Proceedings*, 1989, pp. 700-711.

[13] Jarwala, N. and Yau, C., 1989. A New Framework for Analyzing Test Generation and Diagnosis Algorithms for Wiring Interconnects. *IEEE ITC Proceedings*, 1989.

[14] Yau, C. and Jarwala, N., 1989. A Unified Theory for Designing Optimal Test Generation and Diagnosis Algorithms for Board Interconnects. *IEEE ITC Proceedings*, 1989, pp. 71-77.

[15] Cheng, W., Lewandowski, J. and Wu, E. 1990. Diagnosis for Wiring Interconnects. *IEEE International Test Conference Proceedings*, 1990, pp. 565-571.

[16] Robinson, G. and Deshayes, J. 1990. Interconnect Testing of Boards with Partial Boundary Scan. *IEEE International Test Conference Proceedings*, 1990.

[17] Lien, J. and Breuer, M. 1991. Maximal Diagnosis for Wiring Networks. *IEEE International Test Conference Proceedings*, 1991, pp. 96-105.

[18] Tulloss, R. and Yau, C. 1989. BIST & Boundary-Scan for Board Level Test: Test Program Pseudocode. *European Test Conference Proceedings*, 1989, pp. 106-111.

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

[20] Jong, F. and Heyden, F., 1991. Testing the Integrity of the Boundary Scan Test Infrastructure. *IEEE ITC Proceedings*, 1991, pp. 106-112.

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

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

[23] Maunder, C., 1991. Languages to Support Boundary-Scan Test. *IEEE ITC Proceedings*, 1991, (*panel summary*).

[24] Whetsel, L., 1991. An IEEE 1149.1 Based Logic / Signature Analyzer in a Chip. *IEEE ITC Proceedings*, 1991.

[25] Kritter, S. and Rahaga, T. 1991. Boundary Scan and BIST Compatible IEEE 1149.1: VHDL & Autosynthesis of a SRAM Tester Macrocell and Chip. *European Test Conference Proceedings*, 1991, pp. 17-25.