Proceedings

The European Conference on Design Automation

Brussels, Belgium
March 16–19, 1992

Sponsored by EDAC Association

IEEE Computer Society Press

The Institute of Electrical and Electronics Engineers, Inc.

EDAC 92
pp. 154-158

Automatic Generation of a Single-Chip Solution for Board-Level BIST of Boundary Scan Boards

José M. M. Ferreira, José S. Matos
FEUP (DEEC) / INESC
Rua dos Bragas - 4000 Porto - Portugal

Filipe S. Pinto
INESC
Largo Mompilher, 22 - 4000 Porto - Portugal

Abstract

The automatic generation of a hierarchical self-test architecture for boards with Boundary Scan Test (BST) is described, based on a test processor specifically designed to implement the basic operations required to control the BST infrastructure. An ATPG module generates the ROM containing the test program, allowing a single-chip self-test solution with minimal design-for-testability overhead. The same test processor may be used without internal ROM, when a single-chip solution is not desirable.

1 Introduction

High-complexity components and advanced packaging technologies allow the design of printed circuit boards (PCBs) which pose enormous challenges both for functional and in-circuit test techniques. These challenges led to the adoption of Boundary Scan Test (BST) as a design-for-testability (DFT) methodology able to achieve two main goals:

- To allow structural testing of high-density boards with very limited access to internal nodes.
- To allow functional test of high-complexity components, by providing a gateway to built-in self-test (BIST) functions.

The general acceptance of BST led to its approval as an IEEE standard [1], and contributed to the appearance of a number of components making use of this technology [2].

A number of test controllers supporting BST have been announced [3], [4], [5], [6], either as test controllers for low-cost testers, or for board-level self-test applications. These products do not exploit the hierarchical capabilities of BST, and software support at the automatic test program generation (ATPG) level has not been announced so far.

This paper describes a hierarchical self-test solution for boards with BST, based on a test processor specifically designed to implement the set of basic operations required to control the BST infrastructure. Software support for the proposed solution consists of a board-level ATPG module which automatically generates the ROM containing the test program. The processor itself is a BST component, which allows a hierarchical configuration using the same test processor at different levels.

2 BST: Capabilities, Test Methodology, and Limitations

BST requires every functional pin of each IC to have an associated BS cell, which allows complete controllability and observability of the corresponding electrical node [7]. All these cells are interconnected in a chain which scans the complete PCB, providing a powerful board-level test infrastructure, illustrated in figure 1.

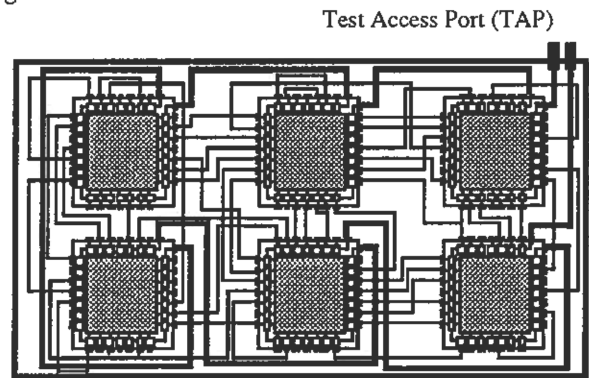

Fig. 1: Board-level BST infrastructure.

This infrastructure is accessible through a *Test Access Port* (TAP), and allows the implementation of straightforward procedures for testing board

interconnects, and for triggering on-chip self-test mechanisms [8].

Interconnect testing allows the detection of open or short faults, which constitute the majority of manufacturing faults on a PCB [9]. Tests for open faults are performed by applying to each path the opposite of the value that would be captured by a floating input. Short fault detection is achieved by applying a set of values which guarantees that every pair of nets will be at complementary logic values at least once. The additional logic required to implement BST on a component provides a gateway to embedded BIST functions, allowing a complete test of the core logic with a minimum of information from the IC manufacturer [10].

These steps should be preceded by a test of the BST infrastructure, which leads to a complete test protocol that may be summarized as follows [11]:

- BST infrastructure test, consisting of initializing the BST logic, shifting of known data through the board BST chain, and checking the device identification registers.
- Interconnect testing, consisting of shifting and applying a set of vectors generated for open and short fault detection.
- Component testing, which will exercise available BIST functions. A set of vectors may also be shifted and applied to the internal logic of components without BIST, or to clusters of non-BST components.

While the characteristics of BST make it an almost ideal solution for testing high-complexity PCBs, two limitations should be referred. Its serial nature prevents real-time functional testing, and there is an intrinsic diagnosis limitation. This limitation occurs because a digital probe (the BS cell) is used to read a frequently analog value, e.g. the result of a short, and also because a fault does not always lead to a deterministic behaviour, e.g. a floating CMOS input.

The limitations referred require the presence of complementary test techniques, but do not affect the complete fault detection capability of BST. This means that this technology provides a solid infrastructure for the development of board-level BIST functions, with a large potential in field maintenance applications (go/no-go test), or which may be used as a valuable aid in production testing.

3 A Test Processor for Board-Level BIST

A protocol for testing BST boards was described in the last section, with a set of steps which may be decomposed in a limited number of basic operations. These operations are responsible for elementary actions, such as initializing the BST logic, forcing a state transition on the BST control logic of each component, shifting bit strings through the BST chain, etc. Precise identification of the required elementary actions allows the definition of an instruction set which should be supported by a test processor specifically designed to test a board through its BST infrastructure, and which is described in tables 1 to 3. This instruction set assumes that a board may contain one or two BST chains (one or two TAPs), and allows synchronization of the BST chain(s) with external test resources through simple handshaking procedures.

Control of the BST infrastructure †	
Procedure	**Instruction**
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**

Table 1: Instructions to control the BST infrastructure.

Although a simple pass/fail indication for the complete set of test vectors will be sufficient for many applications, a pass/fail indication for specific groups of test vectors may be kept internally. This requirement leads to the existence of two sets of eight error flags (one devoted to each board BST chain), and allows a higher level processor to enquire whether a specific type of fault was detected on a specific BST chain.

† N is the value loaded in an internal counter.

Control of internal processor resources	
Procedure	Instruction
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**

Table 2: Instructions to control internal resources.

Test execution synchronization	
Procedure	Instruction
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 3: Instructions for external synchronization.

The instruction set requisites described led to the hardware architecture shown in figure 2, where the *ScanIn* and *ScanOut* blocks interact with the serial path in the board BST chain(s). The *Status* block provides two groups of 8 error flags, and one end-of-test flag.

The processor itself is a BST component, which allows the same test processor to be used at different hierarchical levels, such as illustrated in figure 3. One of the commands supported by the processor BST logic is *Run Board Test*, which allows a higher level processor to trigger a BIST function in each board containing its own test processor. The status block shown in figure 2 is accessible through the processor BST logic, which allows the higher level test processor to collect information concerning the types of faults detected in each BST chain, for each board present. Test program execution will in this case be controlled by the system test clock, which allows complete control of the board-level test processor. Production board testing will in this case be simplified, if an external test equipment replaces the system-level test processor.

A group of six input pins is available as an identification bus in each test processor, which allows the assignment of a specific address to each board present. This identification bus is accessible through the processor BST logic, allowing a higher level processor to perform a system-level check for proper board placement.

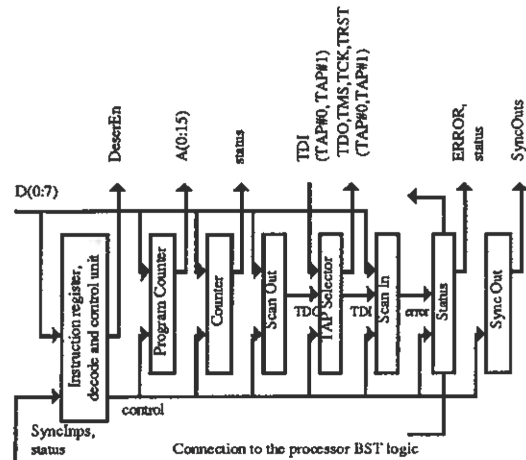

Fig.2: Internal architecture of the test processor.

This architecture provides all the resources required to implement a hierarchical self-test solution for boards with BST, and is complemented by an ATPG module which generates the ROM containing the test program, for each application.

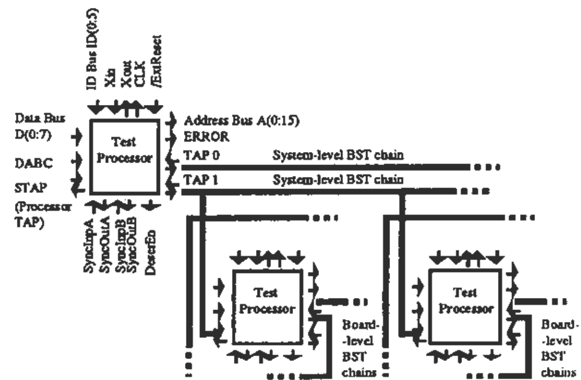

Fig.3: Hierarchical capabilities of the BST processor.

4 The Board-Level ATPG Module

The ATPG module accepts a set of input data structures describing the board interconnects, and the BST infrastructure in each component, and generates a test program which covers the complete test protocol

described in section 2. The information present in these data structures is obtained from a set of input files containing the interconnect description (PCB netlist), and the description of the BST logic in each component (BST data sheets). ATPG for boards with clusters of non-BST components requires additional input files describing the respective test vectors, according to the general flow diagram illustrated in figure 4.

The ATPG module addresses the three main steps described in section 2, which consist of generating the test program for checking the integrity of the board-level BST infrastructure [12], for interconnect testing [13], [14], [15], [16], and for component testing. ATPG for interconnect testing is based on a fault model consisting of open faults, stuck-at faults, and short faults. It is assumed that floating inputs will capture a logic **1**, and that short faults will be of the wired-and type (all inputs will capture a **0**, if any of the shorted outputs is at **0**).

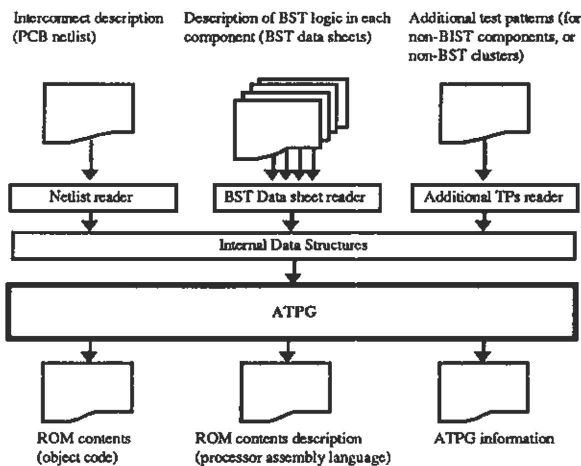

Fig.4: Data flow for the board-level ATPG module.

These technology assumptions define a deterministic behaviour under faulty conditions, which will not be true for certain technologies, e.g. CMOS. However, this fact will not impair the fault detection capability of BST, which is the main requirement for the testing scenarios addressed. Short fault detection is not affected by an eventually random nature of the value captured at input pins, since it is reasonable to assume that all input pins will capture the same value. The short fault will therefore be noticed, even if it does not fit into a deterministic behaviour (wired-and, wired-or, or strong-driver). An open fault may pass undetected if the value captured at a floating input is not predictable, but the probability of such an event is rather low (actually, it is given by $1/2^n$, where n represents the number of test vectors applied).

The information available to the ATPG module does not describe the functionality of the circuit, since the only knowledge required about the core logic of each component should ideally be restricted to a description of how to exercise available BIST functions. This is one of the benefits of BST, but care must be taken that conflicts will not occur due to test vectors enabling more than one driving pin on the same net. This situation will not cause any problems for open or stuck-at fault detection, since only the faulty net has to be driven. However, short fault detection should be done with all nets driven simultaneously, which requires a careful selection of the driving pins to be enabled. The ATPG module performs a depth-first search with backtracking, until it identifies a set of driving pins such that all nets will be driven, with only one driving pin enabled in each net.

The ATPG module outputs two files defining the contents of the ROM to be used with the test processor. One of these files is directly accepted by the ASIC design environment used, which enables the generation of a single-chip self-test solution. The other file describes the test program at assembly language level. An additional output file contains data describing the ATPG process, including the number of test vectors generated, the number of test clock cycles required for complete test program execution, etc.

5 Results and Future Directions

This paper described the automatic generation of a single-chip solution for the self-test of boundary scan boards. The main effort was concentrated on the development of a test processor specifically designed to implement the set of elementary actions required to test a board through its BST infrastructure, and to exploit the hierarchical capabilities of this methodology. This test processor (without internal ROM) was implemented in a 1,5μ CMOS technology, occupying approximately 25 mm^2, in a 68-pin PLCC package.

The development of the ATPG module faced some difficulties, caused by the absence of stable data formats for input data representation. BSDL [17] is now moving towards standardization as a language for describing the BST implementation in each device, but there will still be some time before a common data sheet format is accepted by component manufacturers.

Future directions point towards two main areas: Refinement of the ATPG module, and identification of

alternative applications for the proposed test processor.

The ATPG module will benefit from the standardization of a common data format for the description of component-level BST features. This move towards standardization is taking place under the coordination of a subcommittee created by the IEEE 1149.1 Working Group to investigate the needs and specifications of a description language [17], and which used BSDL version 0.0 as a starting point.

The architecture of the proposed test processor is believed to have reached a stable form, providing the complete set of features required to exploit the potential test capability of board-level boundary scan. The use of an external ROM will broaden the number of boards eligible for BIST inclusion, since a cost/benefit analysis will only recommend a single-chip self-test solution for a limited number of designs.

Alternative applications using the proposed test processor as their main component are being developed. A self-contained BST learning kit will allow a set of features for test program development / debugging, such as single-step execution, breakpoint insertion, etc. Another project concerns the development of a low-cost test / debugging equipment with applications in the area of prototype validation for boards with BST, and also on production testing.

References

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

[2] 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.

[3] 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.

[4] S. Vining, "Tradeoff Decisions Made for a P1149.1 Controller Design," in *Proc. of the IEEE International Test Conference*, August 1989, pp. 47-54.

[5] B. I. Dervisoglu, "Features of a Scan and Clock Resource Chip for Providing Access to Board-Level Test Functions," *Journal of Electronic Testing: Theory and Applications*, Vol. 2, Nº 1, pp. 107-115, March 1991.

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

[7] C. Maunder and R. E. Tulloss, "An Introduction to the Boundary Scan Standard: ANSI/IEEE Std 1149.1," *Journal of Electronic Testing: Theory and Applications*, Vol. 2, Nº 1, pp. 27-42, March 1991.

[8] K. Parker, "The Impact of Boundary Scan on Board Test," *IEEE Design and Test of Computers*, pp. 18-30, August 1989.

[9] F. Jong, J. S. Matos, and J. M. Ferreira, "Boundary Scan Test, Test Methodology, and Fault Modeling," *Journal of Electronic Testing: Theory and Applications*, Vol. 2, Nº 1, pp. 77-88, March 1991.

[10] R. P. van Riessen and H. G. Kerkhoff, "Automatic Test-Specification Generation for Macro-Level BIST Based on the Boundary Scan Standard," in *Proc. of the European Test Conference*, April 1991, pp. 447-453.

[11] R. E. Tulloss and C. Yau, "A Test Program Pseudocode," in C. Maunder and R. Tulloss, "*The Test Access Port and Boundary Scan Architecture*," The IEEE Computer Society Press, ISBN 0-8186-9070-4, pp.97-113.

[12] 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*, October 1991, pp. 106-112.

[13] N. Jarwala and C. Yau, "A New Framework for Analyzing Test Generation and Diagnosis Algorithms for Wiring Interconnects," in *Proc. of the IEEE International Test Conference*, August 1989, pp. 63-70.

[14] C. Yau and N. Jarwala, "A Unified Theory for Designing Optimal Test Generation and Diagnosis Algorithms for Board Interconnects," in *Proc. of the IEEE International Test Conference*, August 1989, pp. 71-77.

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

[16] J.-C. Lien and M. A. Breuer, "Maximal Diagnosis for Wiring Networks," in *Proc. of the IEEE International Test Conference*, October 1991, pp. 96-105.

[17] 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.