International Design Automation Workshop

Moscow, Russia, July 18-21, 1993

AN HDL-BASED APPROACH TO BIST OF 1149.1-COMPATIBLE BOARDS

José M. M. Ferreira[1,2], Manuel G. Gericota[2], José L. Ramalho[2], Gustavo R. Alves[2]

[1] Faculdade de Engenharia (DEEC)
University of Porto
Rua dos Bragas
4000 Porto - PORTUGAL

[2] INESC
Largo Mompilher, 22
4000 Porto - PORTUGAL
Tel. 351-2-321006
Fax: 351-2-318692

Abstract

Boundary scan is now largely accepted as the most promising approach to the testability of highly complex boards, but the number of off-the-shelf BST components available to board-level designers is still rather limited. A proposed solution consisting of a set of board-level testability building blocks is presented in this paper, where an HDL-based design ensures maximum flexibility to the requirements of each board, and a PLD-based implementation ensures low-cost and wide availability.

1. Introduction

The progress in the fields of miniaturisation (surface mount technology, large pin count ICs, etc.) and integration density (due to feature size reduction, and exploited by the availability of highly sophisticated CAD design tools) has made it possible to design very complex printed circuit boards (PCBs), which present very high testability requirements. Boundary Scan design and test [1], [2] is now largely accepted as one of the most promising solutions for this challenge, with an increasing number of off-the-shelf BST components becoming available, and easy-to-use software tools which automate the development of the boundary scan infrastructure for ASIC design [3], [4].

Board-level test, which was the main driving force behind the development of the BST standard, is however still waiting for an integrated family of components able to address three main requirements: the test of non-BST clusters, analog I/O interface, and board-level BIST capability. Proposed solutions for these problems have been published and some components are available [5]-[11], but a much larger offer for board-level designers is still required.

This paper proposes a board-level BIST strategy based on three types of testability building blocks: the interface to non-BST digital I/O nodes, the interface to analog I/O nodes, and a dedicated test processor providing the board-level test capability. It is shown that, by following careful design rules, it is possible to implement all the proposed building blocks in medium-complexity programmable logic devices (PLDs) widely available, therefore providing a low-cost and maximum-flexibility solution for board-level BIST. Moreover, and since these testability blocks were implemented using a simple and powerful hardware design language (HDL), any changes due to specific board requirements can easily be made.

2. Board-Level testability requirements

The number of off-the-shelf BST components replacing frequently used non-BST equivalents is still limited. This restriction makes it very difficult for any board-level design to be 100% BST compatible, except for the rare cases where the designers are allowed to use ASIC technology without restrictions [12]. It is worth mentioning at this point that restrictions are frequently present even when ASIC technology is employed: — minimising the number of ASICs present on a board will normally make it cheaper, and each ASIC should have minimum die area and package size requirements (adding BST should neither represent a significant area overhead, nor make it necessary to choose another package, due to the 4/5 additional pins). The common result is that a board will generally have a BST infrastructure, although providing only limited fault coverage capability [13]-[15].

A need for a set of low-cost and widely available testability building blocks is therefore identified, so that the existing board-level BST resources may be improved and fully exploited. Medium to high-complexity PLDs are now widely available, providing the integration capability to allow single-chip implementations of these testability building blocks. In fact, PLDs are now largely used in low-end ASIC technology applications, and provide two very important advantages over standard-cells or gate-arrays: — immediate prototyping (on a matter of minutes) and maximum flexibility (changes can be made without additional cost). Also, and for small volume productions, shorter time-to-market may be combined with the lower price of factory-programmed parts.

Providing a low-cost and maximum-flexibility solution to improve the testability of BST boards is therefore possible, the first step being to identify board-level test requirements. Interconnects associated to BST pins provide excellent levels of controllability and observability (C&O), which allow straightforward procedures for structural fault detection and diagnosis. However, the low C&O levels associated to those interconnects buried into non-BST clusters can make these areas extremely difficult to test, mainly when two situations are present: — non-BST digital clusters employing high-complexity components, or analog clusters with reduced access through the available BST infrastructure. Two of the main board-level testability requirements may therefore be stated as follows:

- BST access to non-BST digital nodes is required, both to primary I/O pins, and to those pins buried into non-BST clusters. Simple access (EXTEST operating mode) should be provided to primary I/O pins, but more powerful resources should be available for dealing with non-BST clusters: — pseudo-random pattern generation (PRPG) and signature analysis (SA).
- BST access to analog nodes is required. However, and due to the complexity of fully testing an analog cluster, access to these nodes is limited to two basic operations: — capturing the analog values present on the nodes to be observed, and forcing the required analog values on the nodes to be controlled.

Finally, the addition of board-level BIST is only possible if the complete set of low-level TAP (BST Test Access Port) operations to take place in each IC is stored on-board, including all the test vectors used. Testing a board through its BST infrastructure proceeds in three main steps, which consist of testing the BST infrastructure itself, testing the interconnects among the components (including those buried into non-BST clusters), and testing the components (mainly through the activation of component-level BIST functions) [16]. A careful analysis of all the low-level TAP operations which take place in each of these steps leads to the identification of the following three main types: — state transition, where the TMS value defines the next state; application of TCK cycles while TMS is kept at "0" (to execute component BIST functions); and shifting data through the selected registers, which takes place by keeping TMS at "0", except on the last bit to be shifted (in order to step from the Shift state to the Exit1 state). The repeatability of these "standard" low-level TAP operations suggests that a simple RISC processor, with an instruction set specifically designed to implement the three types of operations identified, would constitute an important board-level testability requirement, if board-level BIST is to be supported [17]. This final requirement may therefore be specified as follows:

- Board-level BIST should be supported by a dedicated test processor, with an instruction set designed to optimise the three main types of low-level TAP operations which take place when testing a board. The programming model of this BIST processor must be used by an automatic test program generation (ATPG) tool to produce the test code to be executed. Notice however that it is not possible to fully automate the task of this ATPG tool, since non-BST (digital and analog) clusters, as well as specific conditions inherent to each design (like illegal conditions leading to bus conflicts), will always be present in some degree.

Proposed solutions for each of these board-level testability requirements will now be presented.

3. Board-level testability building blocks

The board-level testability requirements presented in the previous section led to the development of four types of testability building blocks, which will now be described.

3.1. The interface to non-BST digital I/O nodes

Non-BST digital I/O nodes present on a board may be divided into two main types: — primary I/O pins, and pins belonging to non-BST clusters (either in its boundary, or buried). Different solutions are considered for these two situations.

3.1.1. PRIMARY I/O PINS

Faults present on primary I/O pins can normally be detected only if test resources external to the board are present, although in some cases this test might be accomplished by synchronising the BST chains present on boards connected by a common backplane (system-level test) [18]. However, and considering a production test scenario, external test resources must be used to detect faults present on interconnects with primary I/O pins. Parallel test channels synchronised with the on-board BIST processor might be used, but a simpler solution may be found by extending the board-level BST chain with a set of cells controlled by the on-board BIST processor, such as illustrated in figure 1.

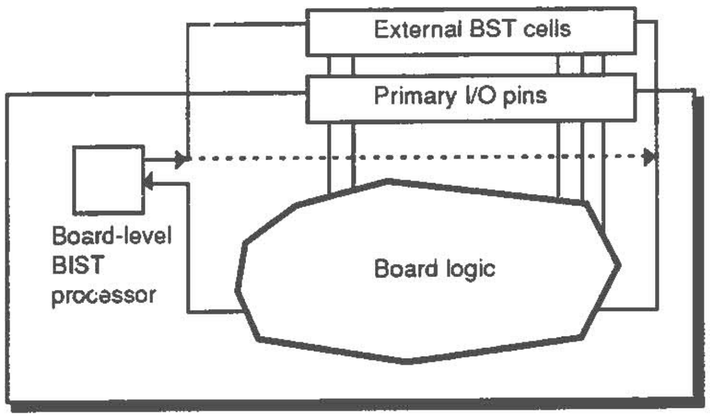


Fig. 1: Test of primary I/O pins under control of the on-board BIST processor.

This situation will probably not be applicable in every test scenario (like on most field-maintenance operations), and a different test program must be used, but the advantages of extending the BST infrastructure to test the primary I/O pins are worth the development of a simple component with no core logic, containing only the external BST cells required. The block diagram of this component is shown in figure 2.

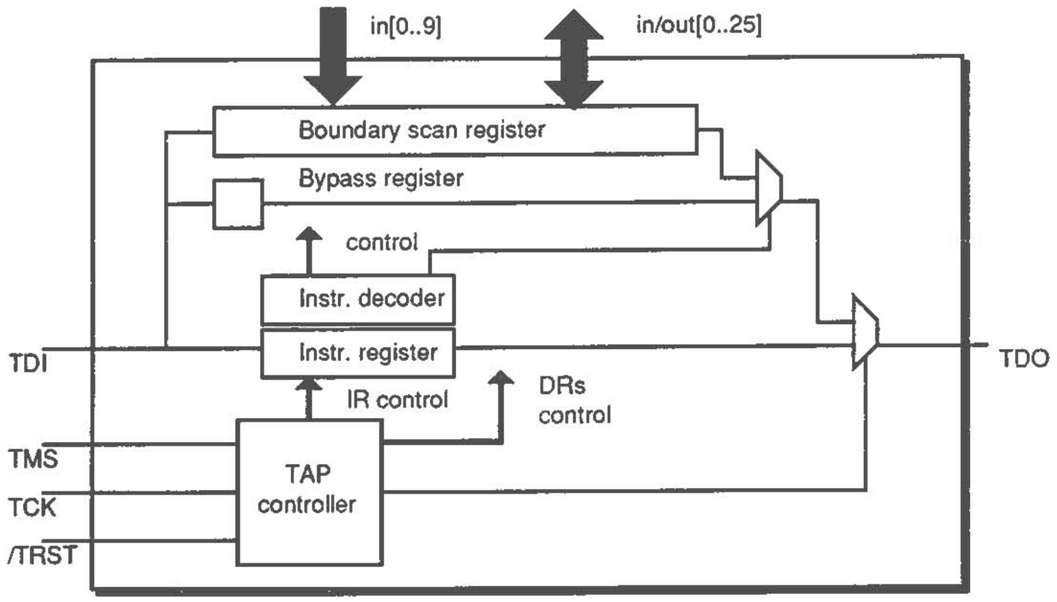
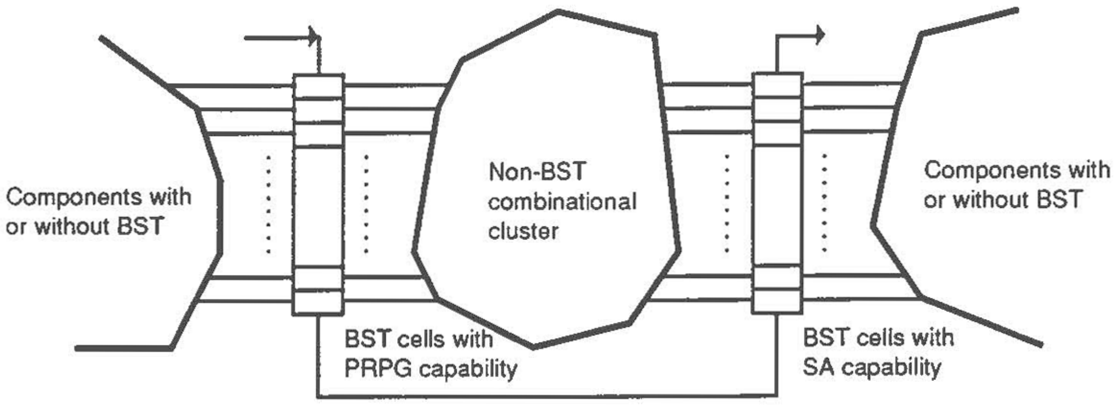


Fig. 2: Block diagram of the primary I/O pin test component.

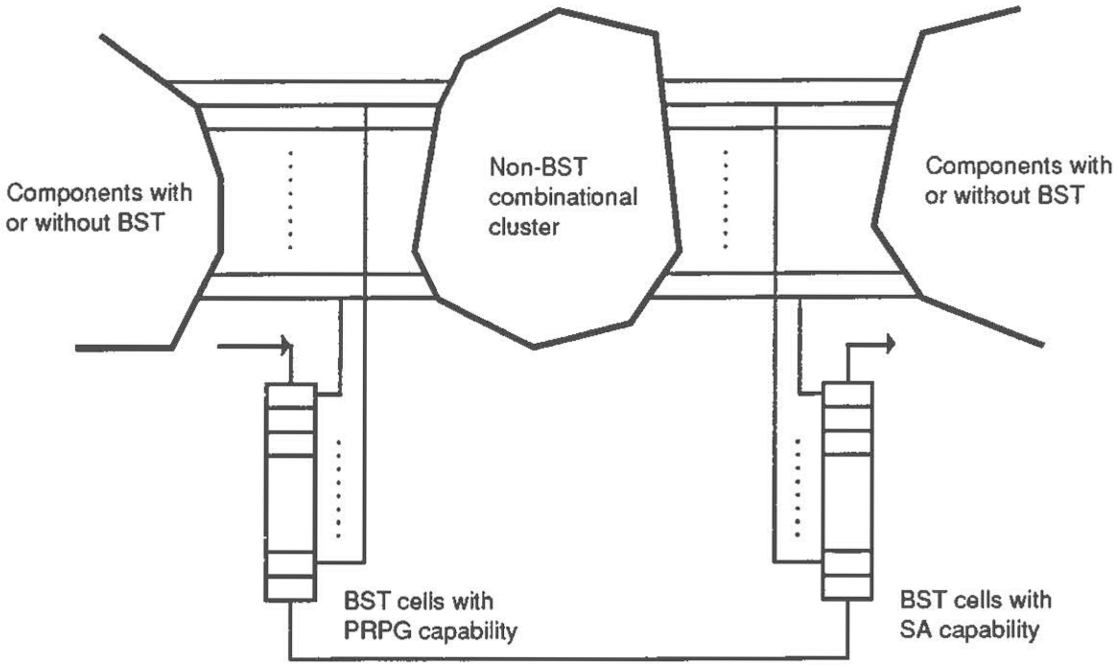
The boundary scan register of this component consists of 10 input pins and 26 bidirectional pins (with individual tristate control) and the instructions supported conform to the IEEE 1149.1 standard requirements (EXTEST, SAMPLE / PRELOAD and BYPASS). Complete C&O over the primary I/O pins may be achieved by simply cascading the required number of these components. A similar solution might be achieved by using commercially available BST octals, but these components would not allow individual tristate control of each output, which may be required.

3.1.2. PINS BELONGING TO NON-BST CLUSTERS

Complex non-BST clusters may have to be tested by in-circuit test equipment, although there are cases where the surrounding BST infrastructure may instead be used (virtual cluster testing). An example may be found on combinational clusters, which may efficiently be tested by PRPG and SA techniques. Components supporting these techniques are normally inserted into the signal flow path, such as illustrated in fig.3.(a). This is an application where a PLD would not be recommended, both because of the long propagation times associated with PLD technologies, but also because optimised low-cost components with PRPG and SA capability, or even with more sophisticated operating modes, are now already available [8], [10]. Alternative applications might however be considered, either when clusters are directly connected to primary I/O pins, or when parallel connections to cluster nodes are required, such as illustrated in figure 3.(b).


Components with or without BST
Non-BST combinational cluster
Components with or without BST
BST cells with PRPG capability
BST cells with SA capability

(a) PRPG and SA structures inserted into the signal flow path.


Components with or without BST
Non-BST combinational cluster
Components with or without BST
BST cells with PRPG capability
BST cells with SA capability

(b) PRPG and SA structures inserted in parallel with the signal flow path.

Fig. 3: Virtual cluster testing by PRPG and SA techniques.

In these cases, individual tristate control or programmable-length LFSRs might be useful, features which are not supported by commercially available components. Programmable-length LFSRs might specially be of use when guarding values are to be applied through those bits not used for PRPG. These requirements led to the development of a programmable-length LFSR PLD, providing both PRPG and SA, with individual tristate control, and where those bits not used for PRPG will keep their initial value. An internal control register may be selected by a dedicated instruction, and loaded with a 4-bit word which defines the number of bits required to be non-PRPG outputs with individual tristate control (between 0 and 15). Since a total of 20 output pins exist, the length of the LFSR will be given by 20-(CR), where (CR) represents the value loaded into the control register. The block diagram of this PLD is illustrated in figure 4.

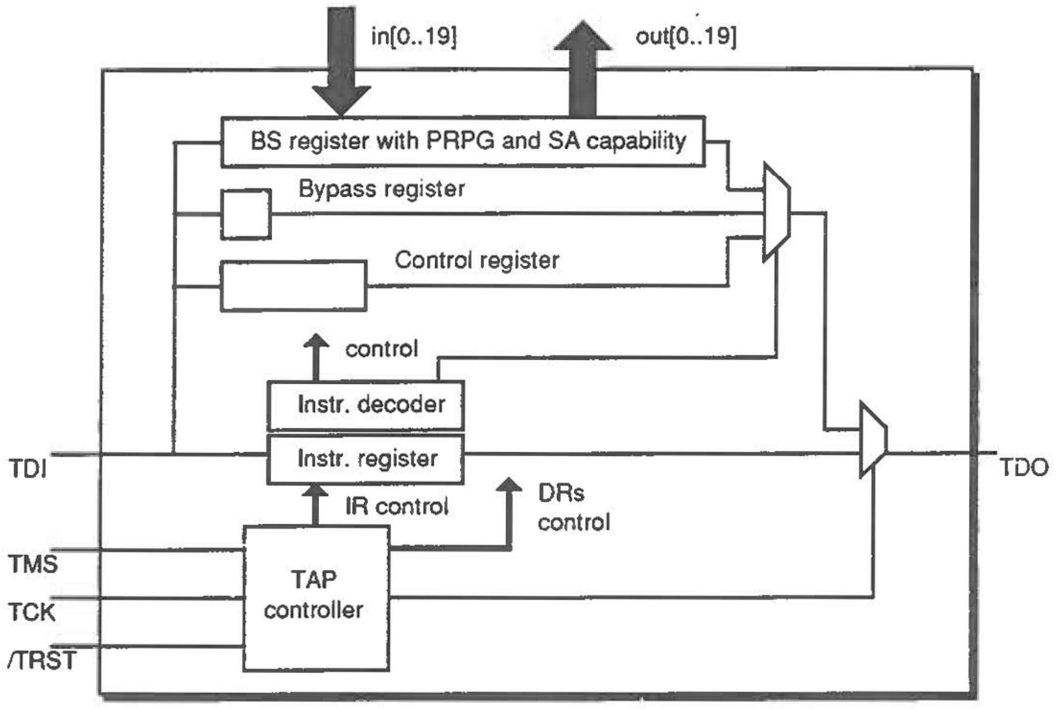


Fig. 4: Block diagram of the programmable-length LFSR PLD.

3.2. The interface to analog I/O nodes

The main goal behind the development of the BST technology was to provide structural testing of high-complexity digital boards. Functional test, or structural test of analog circuits, are therefore areas where the BST technology faces serious limitations [19]. An interface to analog I/O nodes may therefore be very useful, even if restricted to simple low-speed test operations [20].

In order not to cause delays, distortion, or frequency response limitations on the analog signals, no analog multiplexers should be inserted into the signal flow path. Capture operations would therefore be possible on any desired analog node, but the only analog nodes to be controlled could only consist of primary input nodes, through a set-up similar to the one shown in figure 1. However, if it is acceptable to insert analog multiplexers, the solution illustrated in figure 5 may be implemented.

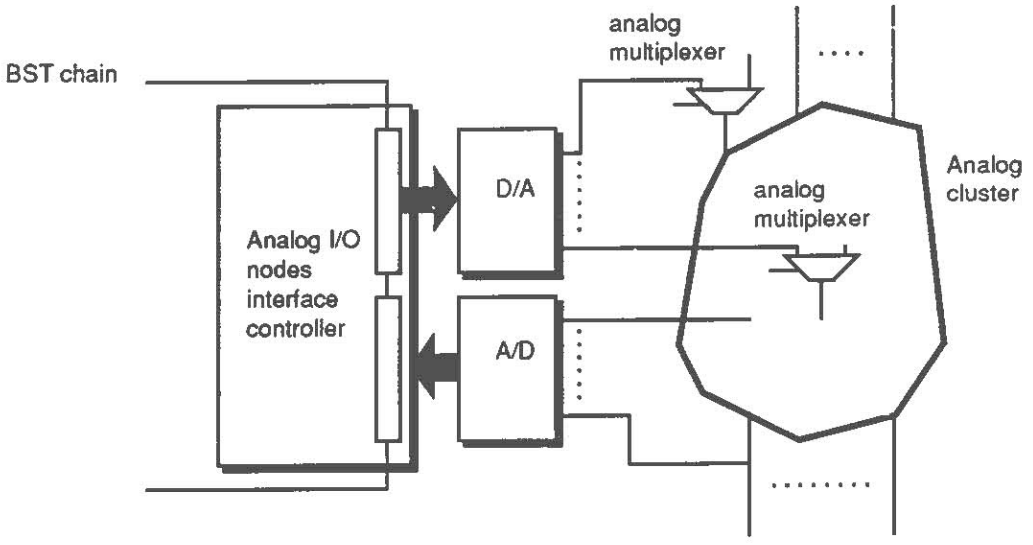


Fig. 5: The interface to analog I/O nodes.

The analog I/O nodes interface controller provides an output pin for controlling the analog multiplexers shown in figure 5, and allows access to 16 analog inputs and 16 analog outputs. The block diagram of this interface controller may be represented as shown in figure 6.

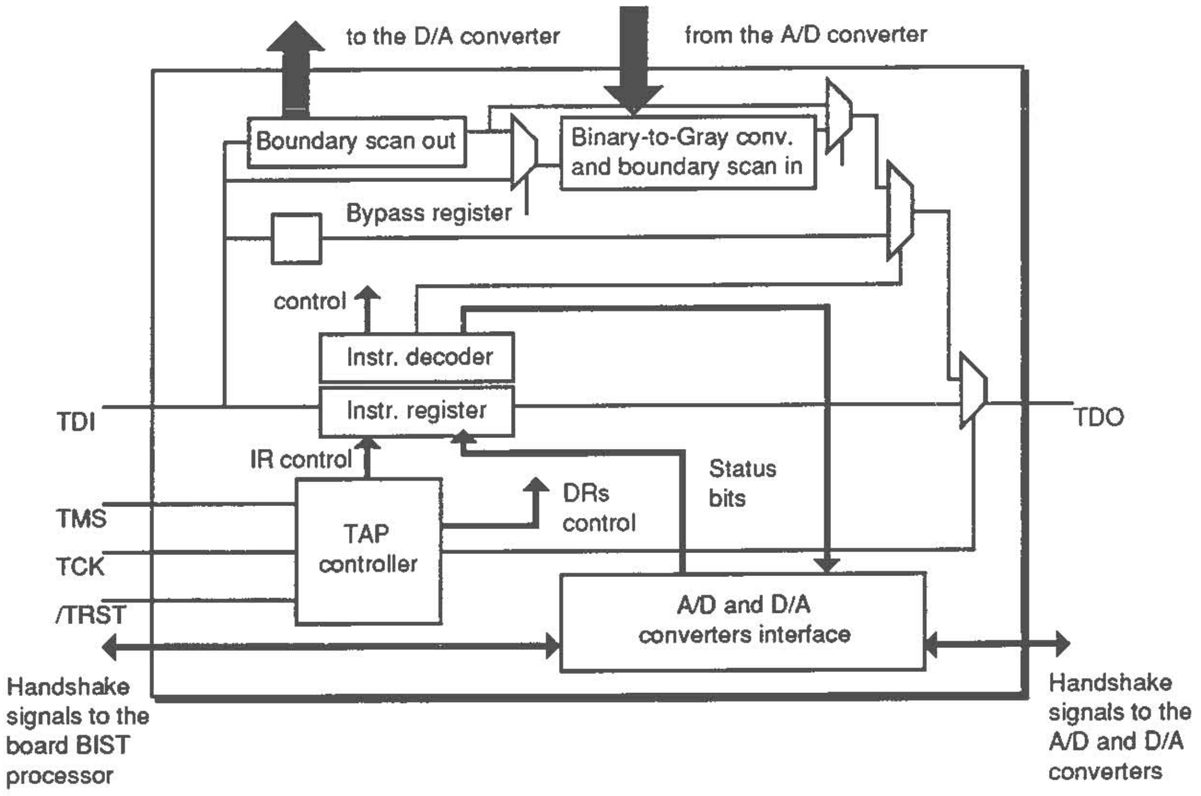


Fig. 6: Block diagram of the analog I/O nodes interface controller.

An 11-bit instruction register allows A/D or D/A conversions to take place individually on the selected channels (which may be different for A/D and D/A operations). The A/D converters used are assumed to be of the successive-approximation type, and the end of conversion state may be checked by examining the bits shifted out of the instruction register following a Capture-IR operation.

Capture and compare operations on analog I/O nodes assume that there is an interval on which the captured analog value is considered correct, meaning that some type of mask must be used to specify the acceptable range at the output of the A/D converter. However, and since adjacent binary codes may exhibit changes on as many as *every* bit (consider for example the codes corresponding to decimals 127 and 128), a code conversion operation must be performed, so that a mask can be used to specify the accepted deviation. A simple solution consists of performing a binary to Gray code conversion, which guarantees that successive codes do not differ in more than one bit position. Allowing a four-code acceptable range may therefore be accomplished by using a mask generated by ex-noring the two codes adjacent to the expected value. As an example, and if the expected Gray code word is 00011100, the mask word is $/(00011101 \oplus 00010100)=11110110$. Use of this mask for comparing the Gray code equivalent of the A/D converter output will correspond to an acceptable range defined by 1111X11X (X=don't care). Since the binary to Gray code conversion is achieved by ex-oring each bit to its left neighbour (binary I_n, I_{n-1}, ..., I_0 corresponds to Gray I_n, $I_n \oplus I_{n-1}$, ..., $I_1 \oplus I_0$), this operation is implemented by simply adding an ex-or to the serial input of the BST cells connected to the A/D converter output.

3.3. The board-level BIST processor

The board-level requirements for the BIST processor led to an optimised instruction set, which allows a straightforward specification of all the low-level TAP operations required for each step of the test sequence. The complete instruction set is described in table 1, including those instructions which do not directly represent TAP operations.

The block diagram of the BIST processor is shown in figure 7. Notice that a TAP selector block allows the internal processor resources to be multiplexed by two board BST chains, and that a 20-bit program counter is able to address test programs with sizes up to 1 Mbyte (test of non-BST clusters without PRPG and SA may produce large test programs).

	TAP operations
SELTAP0, SELTAP1	Selects the BST chain to be controlled by the following instructions.
TRST	Forces an asynchronous reset through the /TRST output of the selected BST chain.
NSHF	N bits will be shifted into the BST chain. Bits shifted out of the BST chain are not compared. N represents the contents of the internal 16 bit counter.
NSHFCP	N bits will be shifted into the BST chain. Bits shifted out of the BST chain are compared with their expected value. Mask bits are used to discard don't care bits. N represents the contents of the internal 16 bit counter.
TMS0, TMS1	Forces a state transition in the internal BST logic of each component, in the selected BST chain.
NTCK	Applies N test clock cycles, while keeping TMS at "0". N represents the contents of the internal 24 bit counter.
	Internal control and synchronisation
LD C16, N	Loads the internal 16-bit counter with the number of test clock (TCK) cycles to be applied.
LD C24, N	Loads the internal 24-bit counter with the number of test clock (TCK) cycles to be applied.
JPE Address **JPNE Address**	Conditional jumps based on the state of the internal error flag.
SS0, SS1	Forces a logical value (0,1) on the synchronism output.
WS0, WS1	Waits for a logical value (0,1) on the synchronism input.
HALT	Terminates test program execution.

Table 1: Instruction set supported by the board-level BIST processor.

An output pin will be active for each TCK cycle where a shift and compare operation takes place (*DeserEn*, in figure 7). This pin may be used to enable an external deserialiser, allowing the results shifted out of the BST chains to be stored in off-board memory, whenever diagnosis operations are required. Three additional output pins provide information on the internal state of the processor: — *end of test*, indicating that test program execution is complete; *error*, indicating if one or more faults were detected; and *SelTAP*, indicating which of the two TAPs supported by the processor is active. Synchronism inputs and outputs, directly controlled by the corresponding instructions referred in table 1, allow the implementation of simple handshake protocols with other test resources (for example, with the start of conversion and end of conversion signals through the analog I/O nodes interface controller).

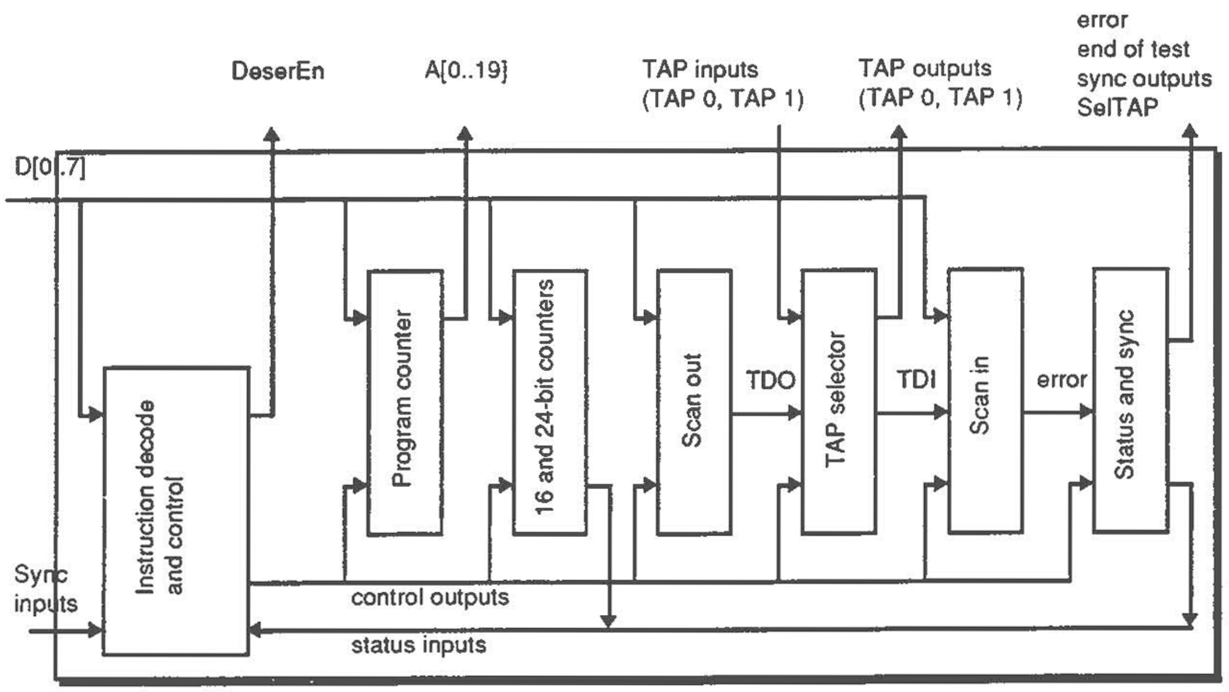


Fig. 7: Block diagram of the board-level BIST processor.

An ATPG tool running on a 486-PC was also developed, which partially automates the task of generating the test program for this controller. This tool reads a set of input files containing board-level structural information, a description of the BST infrastructure present in each component, and a description of existing non-BST clusters (including the identification of the surrounding BST cells, and possibly of externally-generated, deterministic test vectors). The test code, specified in terms of the instruction set presented in table 1, is then generated. It consists of test program segments for checking the integrity of the board-level chains, for interconnect fault detection (both for full-BST interconnects and for cluster interconnects), and for testing the components present on the board.

4. Implementation and application examples

With the exception of the interface to analog I/O nodes, each testability building block described in the previous sections was successfully implemented on one 5128 Altera PLD. This component is a 68-pin medium-complexity device (128 macrocells). The analog I/O nodes interface controller, represented in figure 6, was implemented on a smaller 5064 PLD (44 pins, 64 macrocells). Every component was specified using the Altera hardware design language (AHDL), which proved to allow fast specification and debugging. Careful design rules had however to be observed, since usage of the internal resources had to be extremely optimised (macrocell usage was between 95% and 100%, with 97% average).

Several application examples were used to validate the set of testability building blocks developed. One of these examples is shown in figure 8, and consists of a small board with two BST chains and two simple non-BST digital clusters. The test set-up for this board included one BIST processor responsible for controlling both BST chains. Primary I/O pins were interfaced through a chain of external BST cells connected to TAP 1, such as illustrated in figure 1.

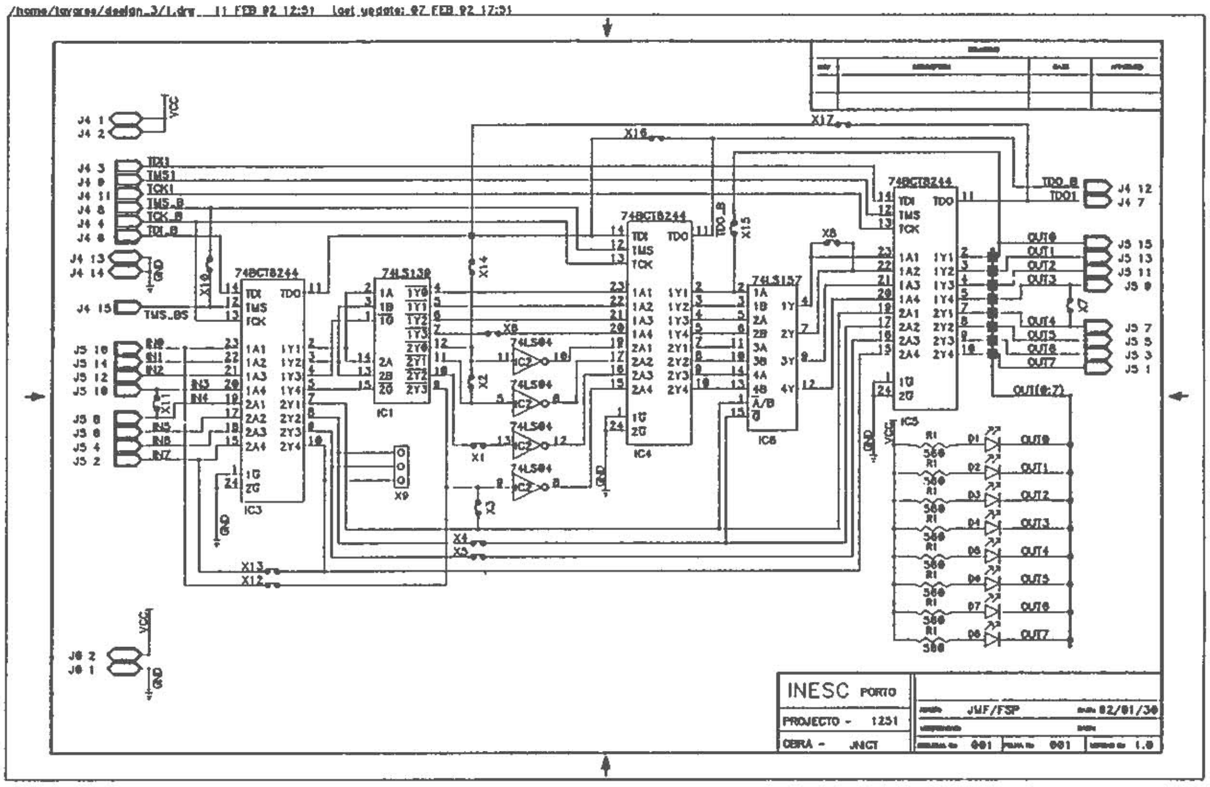


Fig. 8: Circuit schematic of an application example.

The test program for this example was generated by the ATPG tool referred in the previous section, and occupied approximately 1 Kbyte (including the test vectors externally generated for the two non-BST clusters present). This test program may be illustrated by the following segment, which addresses the detection of open faults.

```
00069 02 00 24      ld      c16,36 ; Length of BS chain 0
0006C 04            nshf           ; Shift in vector # 1
0006D 00            .db     $00
0006E 00            .db     $00
0006F 00            .db     $00
00070 00            .db     $00
00071 00            .db     $00
00072 01            tms1           ; Go to Update DR
00073
00073 1B            seltap1        ; Switch to TAP 1
00074
00074 01            tms1           ; Go to Select DR Scan
00075 00            tms0           ; Go to Capture DR
00076 00            tms0           ; Go to Shift DR
00077
00077 02 00 52      ld      c16,82 ; Length of BS chain 1 + ext. prim. I/O PLDs
0007A 04            nshf           ; Shift in vector # 1
```

```
0007B FE            .db    $fe
0007C 01            .db    $01
0007D 00            .db    $00
0007E 00            .db    $00
0007F 00            .db    $00
00080 00            .db    $00
00081 00            .db    $00
00082 00            .db    $00
00083 00            .db    $00
00084 00            .db    $00
00085 00            .db    $00
00086 01            tms1          ; Go to Update DR
00087
00087 01            tms1          ; Go to Select DR Scan
00088 00            tms0          ; Go to Capture DR
00089 00            tms0          ; Go to Shift DR
0008A
0008A 1A            seltap0       ; Switch to TAP 0
0008B
0008B 01            tms1          ; Go to Select DR Scan
0008C 00            tms0          ; Go to Capture DR
0008D 00            tms0          ; Go to Shift DR
0008E
0008E 02 00 24      ld     c16,36 ; Length of BS chain 0
00091 05            nshfcp        ; Shift in vector # 2
00092 00 FF 00      .db    $00,$ff,$00
00095 00 FF 00      .db    $00,$ff,$00
00098 0C FC 03      .db    $0c,$fc,$03
0009B 00 03 FC      .db    $00,$03,$fc
0009E 00 00 0F      .db    $00,$00,$0f
000A1 06 00 05 A7   jpe    theend ; stop the test if a fault is found
000A5 01            tms1          ; Go to Update DR
000A6
000A6 1B            seltap1       ; Switch to TAP 1
000A7
000A7 02 00 52      ld     c16,82 ; Length of BS chain 1 + ext. prim. I/O PLDs
000AA 05            nshfcp        ; Shift in vector # 2
000AB FE FF 00      .db    $fe,$ff,$00
000AE 01 FF 00      .db    $01,$ff,$00
000B1 00 FF 00      .db    $00,$ff,$00
000B4 F8 FF 00      .db    $f8,$ff,$00
000B7 07 FF 00      .db    $07,$ff,$00
000BA 00 FF 00      .db    $00,$ff,$00
000BD 00 7F 80      .db    $00,$7f,$80
000C0 00 80 7F      .db    $00,$80,$7f
000C3 FF FF 00      .db    $ff,$ff,$00
000C6 00 FC 03      .db    $00,$fc,$03
000C9 00 00 03      .db    $00,$00,$03
000CC 06 00 05 A7   jpe    theend ; stop the test if a fault is found
000D0 01            tms1          ; Go to Update DR
000D1
000D1 01            tms1          ; Go to Select DR Scan
000D2 00            tms0          ; Go to Capture DR
000D3 00            tms0          ; Go to Shift DR
000D4
000D4 1A            seltap0       ; Switch to TAP 0
```

Since an 8-bit data bus was specified, the data to be shifted into the BST chains, the expected results, and the mask information, are byte-interleaved in memory. The operand of the NSHFCP instructions shown above therefore consists of three-byte blocks, each with a first byte of data to be shifted, a second byte with expected results, and a third byte with mask information.

5. Conclusion

A set of board-level testability requirements were identified in order to overcome the limitations caused by restricted availability of off-the-shelf BST components. Medium-complexity PLDs were used to implement proposed solutions to these requirements, therefore providing a low-cost and maximum-flexibility solution to this problem.

Fast prototyping (minutes) is a key point for flexibility, since an unrestricted number of changes can be made. This is specially important if we consider that every component was specified using an easy-to-learn hardware design language, which means that any changes can be made by simply editing the corresponding text file. Optimised solutions are therefore easy to implement, providing a straightforward approach to such modifications as changing the ratio of input to bidirectional pins required in the component interfacing non-BST digital I/O nodes, or extending the resolution in analog capture operations to a higher number of bits (12, for example). Finally, and if small volume productions are envisaged, the choice of PLD technology will still provide two additional benefits: — the lower price of pre-programmed parts, and reduced time-to-market periods.

The complete set of PLD specification files are available by public domain ftp, by connecting to ftp.inescn.pt (use anonymous as username, and your e-mail address as password), and moving to a directory called pub/doc/dftplds. Complete specifications and examples are also available by contacting any of the authors at e-mail address jmmf@porto.inescn.pt.

References

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

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

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

[4] Muris, M. 1990. Integrating Boundary Scan Test Into an ASIC Design Flow. *IEEE International Test Conference Proceedings,* 1990, pp. 472-477.

[5] Bruce, W., Gallup, M., Giles, G. and Munns, T. 1991. Implementing 1149.1 on CMOS Microprocessors. *IEEE International Test Conference Proceedings,* 1991, pp. 879-886.

[6] Matos, J. S., Pinto, F. S. and Ferreira, J. M. 1992. A Boundary Scan Test Controller for Hierarchical BIST. *IEEE International Test Conference Proceedings*, 1992, pp. 217-223.

[7] Jarwala, N., and Yau, C. W. 1991a. The Boundary-Scan Master: Architecture and Implementation. *European Test Conference Proceedings*, 1991, pp. 1-10.

[8] 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.

[9] Raghavachari, P. 1991. Circuit Pack BIST from System to Factory - The MCERT Chip. *IEEE International Test Conference Proceedings*, 1991, pp. 641-648.

[10] Whetsel, L. 1991. An IEEE 1149.1 Based Logic / Signature Analyzer in a Chip. *IEEE International Test Conference Proceedings*, 1991, pp. 869-878.

[11] Whetsel, L. 1992. A Proposed Method of Accessing 1149.1 in a Backplane Environment. *IEEE International Test Conference Proceedings*, 1992, pp. 206-216.

[12] Daniel, W. 1992. Design Verification of a High Density Computer Using IEEE 1149.1. *IEEE International Test Conference Proceedings*, 1992, pp. 84-90.

[13] Hansen, P. 1990. Taking Advantage of Boundary-Scan in Loaded-Board Testing. In Maunder, C. et al. *The Test Access Port and Boundary Scan Architecture.* The IEEE Computer Society Press. ISBN 0-8186-9070-4, pp. 81-96.

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

[15] Robinson, G. D. and Deshayes, J. G. 1990. Interconnect Testing of Boards with Partial Boundary Scan. *IEEE International Test Conference Proceedings*, 1990, pp. 572-581.

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

[17] Ferreira, J. M., Matos, J. S. and Pinto, F. S. 1992. Automatic Generation of a Single-Chip Solution for Board-Level BIST of Boundary Scan Boards. *European Design Automation Conference Proceedings,* March 1992, pp. 154-158.

[18] Whetsel, L. 1992. A Proposed Method of Accessing 1149.1 in a Backplane Environment. *IEEE International Test Conference Proceedings*, 1992, pp. 206-216.

[19] Fleming, P. 1990. Applications of the IEEE Std 1149.1: An Overview. In Maunder, C. et al. *The Test Access Port and Boundary Scan Architecture.* The IEEE Computer Society Press. ISBN 0-8186-9070-4, pp. 129-140.

[20] Hirzer, J. 1990. Testing Mixed Analog/Digital ICs. In Maunder, C. et al. *The Test Access Port and Boundary Scan Architecture.* The IEEE Computer Society Press. ISBN 0-8186-9070-4, pp. 199-204.