New Built-In Self-Test Boundary Scan Architectures for Digital Integrated Circuits in Industrial Applications

Mohamed H. El-Mahlawy
Electrical Eng Dept.
Faculty of Engineering and Technology
Future University in Egypt

Sherif Anas
Research and Development of Electronic Factory
Arab Industrialization Authority

Winston Waller
University of Kent, Canterbury,
Kent, CT2 7NT, UK

ABSTRACT
In this paper, the incorporation of pseudo-exhaustive built-in self-test capabilities into the boundary scan (BS) architecture is presented. The Boundary Scan Register (BSR) input cells are configured as a test pattern generator (TPG), and the BSR input and output cells are configured as a test response compactor (TRC) in the BIST mode. Instructions for both BS and BIST process are proposed that enable the test access port controller (TAPC) to control the BS and BIST process. The presented design supports the BIST of the target chip for both the cascaded and non-cascaded input and output cells of the BSR. In the register transfer level (RTL), the insertion of segmentation cells in the case of pseudo-exhaustive testing (PET) may affect the system timing due to the unequal sequential depth in the BIST mode, so it is required to insert delay flip-flops which add significant area overhead and degrade circuit performance. In addition, transferring every flip-flop into BIST flip-flop adds area overhead and degrades circuit performance in the normal mode of the chip. To compensate these problems, a proposed design that converts the presented sequential block into the combinational block (combinational equivalent). The incorporation of BIST capabilities into the boundary scan architecture with this solution is presented. Finally, a complete example for BIST (Built-In Self-Test) boundary scan architecture and 16-bit parallel-pipelined multiplier as the CUT is presented. The simulation and then design download are presented on the field programmable gate array (FPGA) chip. The hardware implementation using the interfacing through the personal computer as a master controller controls the test circuitry from the TAPC as a slave controller.

Keywords
Built-In Self-Test (BIST), Testing of digital circuits, Boundary scan, Design for testability, BIST for boundary scan, Pseudo-exhaustive testing.

1. INTRODUCTION
Advances in very large scale integration (VLSI) technology have led to the fabrication of chips that contain a very large number of transistors, integrated on a single chip. The cost of testing such devices increases with complexity so the incorporation of BIST capability inside a chip is increasingly desirable. The BIST strategy requires hardware overhead to incorporate a TPG, a TRC, and a BIST controller into the system (core) logic [1-13]. In addition, the use of the VLSI chips in advanced fine-pitch packages (e.g. Ball Grid Array (BGA)) prevents the physical probing of the target tested chip to access its pins. Therefore, the boundary scan (BS) technique, formally known as IEEE-1149.1 Standard, offers a convenient alternative to physical probing by effectively migrating the test probe circuitry into the chip [2, 14-20]. It enables a non-contact method of accessing chip pins for testing. The block diagram of the BS architecture is shown in Fig. 1. The basic architecture of this standard is incorporated at the chip level and essentially consists of a protocol by which various test functions can be carried out. The basic concept behind BS is to introduce a scan cell situated just inside the chip input/output (I/O) pad ring at each I/O pin [2]. The BS cells are interconnected to form a scan chain. During normal system operation, these cells are transparent, passing the I/O signals unaltered. However, during the test mode, the logical connection from the I/O pads to the system logic is broken and signals may be injected and sampled via these cells [2, 19-20].

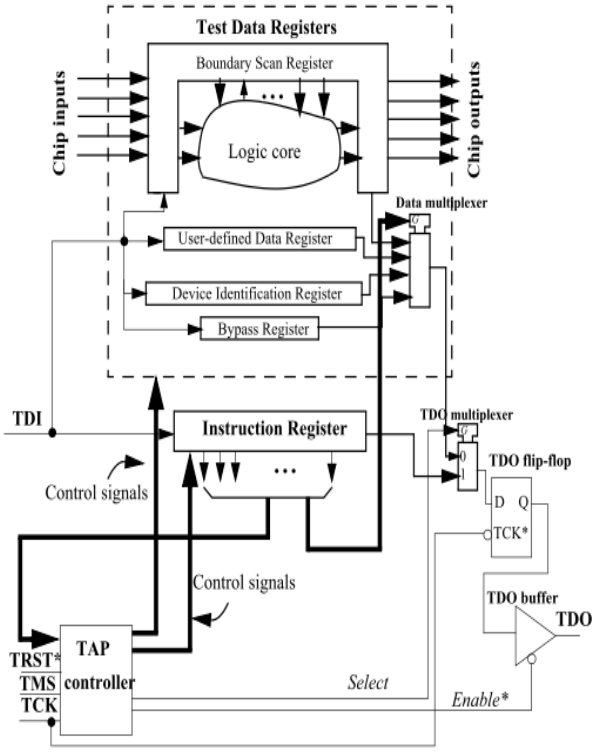

Fig 1: Block diagram of the boundary scan in the chip level.

Using an appropriate set of the BS test patterns, most of the interconnections on a circuit board can be checked for continuity and short circuits. The IEEE Standard 1149.1 defines an interface for the BS at device and assembled circuit board level. It defines four (or optionally, five) pins, through which the BS test features, are controlled, forming the test access port (TAP). Two of them (test clock, *TCK*, and test mode select, *TMS*) are used to control the protocol, while the remaining two pins (test data input, *TDI*, and test data output, *TDO*) are employed to serially shift data into and out of the

chip. The optional pin (test reset input, *TRST**) asynchronously forces the test logic into its reset state to ensure normal system operation at power up when the test logic is not in use.

Due to the cost of testing VLSI chips, the incorporation of BIST capability inside a chip is increasingly desirable [21-26]. It is required to build a TPG in the BIST mode as a stimulus for chip inputs, and a Multi-Input Shift Register (MISR) in the BIST mode as a test response compactor (TRC) for chip outputs. The BIST controller controls the BIST process. The BIST requires hardware overhead to incorporate a TPG, a TRC, and a BIST controller into the system (core) logic. The incorporation of BIST capabilities into the BS architecture, defined in the IEEE-1149.1 standard has been investigated [14, 18]. In addition, the BSR input cells have been configured to operate as a TPG in the BIST mode. The BSR output cells have been configured to operate as an MISR in the BIST mode. For the register transfer level (RTL), the insertion of segmentation cells in the case of pseudo-exhaustive testing (PET) [9-10, 27-30] may affect the system timing due to the unequal sequential depth in the BIST mode, so it is required to insert delay flip-flops which add significant area overhead and degrade circuit performance. In addition, transferring every flip-flop into a BIST flip-flop adds area overhead and degrades circuit performance in the normal mode of the chip.

In this paper, the incorporation of BIST capabilities into the BS architecture to support the BIST for either the cascaded or non-cascaded BSR input and output cells is presented. The TAPC controls the BIST process. New instructions for BIST process are proposed. This configuration supports the BIST for both the Built-In Logic Block Observer (BILBO) register [2, 31-32] and the RTL. The problem of insertion of delay flip-flops as the added area overhead and degrade circuit performance in the normal mode of the chip. To compensate this problem, the design solution is presented to convert the problem into one combinational block (combinational equivalent). In addition, the incorporation of BIST capabilities into the boundary scan architecture with this solution is presented. Using the inherent BIST capabilities in the boundary scan, new boundary scan architecture for BIST is introduced. The steps of the BIST process are implemented based on the developed new instructions.

This paper is organized in seven sections. The presented section describes an introduction to the BIST and the BS architecture for testing of digital integrated circuits. Section 2 presents the basic concepts of the BIST of the PET. Section 3 presents the design of the BIST boundary scan architecture. Section 4 presents the single-test clock approach. Section 5 presents the complete testable 16-bit parallel pipelined multiplier with BIST boundary scan architecture. Section 6 illustrates the experimental results and finally the last section concludes the presented work.

2. BASIC CONCEPTS OF THE PSEDO-EXHOSTIVE BIST TESTING

Traditionally, the circuit under test (CUT) is tested by the automatic test equipment (ATE), which can store the applied test patterns and the expected test response of the CUT. In the Built-In Self-Test (BIST) mode, extra circuitry is added around the original CUT to test itself. Systems, boards, and chips can be tested in this mode. This form of testing is also applicable at the manufacturing, field, depot, and operational levels. Often, testing is carried out using on-chip or on-board TPGs and TRCs. The BIST incorporates a TPG such as autonomous linear feedback shift register (ALFSR) shown in Fig. 2, a TRC such as an LFSR in the serial BIST shown in Fig. 3 and the multi-input shift-register (MISR) in the parallel BIST shown in Fig. 4, and a BIST controller into the system (core) logic to realize self-test operations. Fig. 2, Fig. 3, and Fig. 4 have c_i's as binary constants, c_i = 1 implies that a connection exists, while c_i = 0 implies that there is no connection.

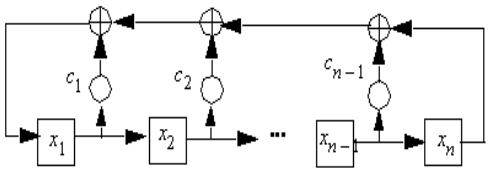

Fig. 2: ALFSR as the TPG.

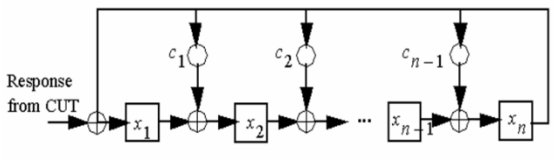

Fig. 3: LFSR as the TRC in the serial BIST

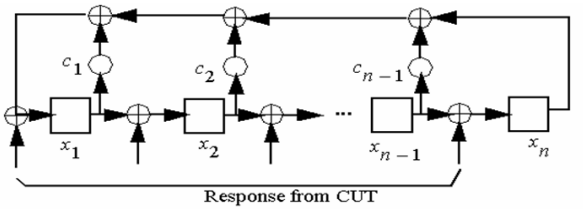

Fig. 4: MISR as the TRC in the parallel BIST.

The fundamental trade-off between time and hardware has created two basic BIST execution options: *parallel BIST or test-per-clock technique* and *serial BIST or test-per-scan approach* [2]. In test-per-clock BIST shown in Fig. 5, test patterns are applied from the TPG and test responses are captured in the TRC (MISR) every clock cycle. Notice that the scheme introduces performance degradation due to the presence of the multiplexer between the primary inputs and the CUT.

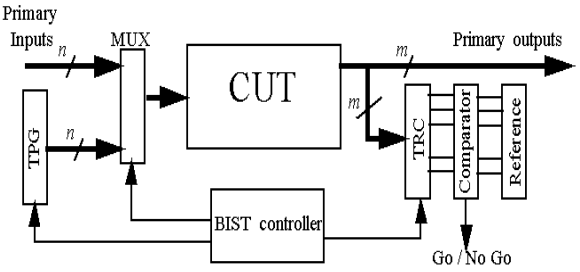

Fig. 5: Basic Block diagram of the parallel BIST.

In test-per-scan BIST shown in Fig. 6, test patterns are shifted into serial scan path, applied to the CUT, and test responses are subsequently captured in the scan flip-flops and shifted out to the TRC while a new test is being shifted in. Clearly, the serial BIST is much slower in applying tests than the parallel BIST. On the other hand, the serial BIST takes advantage of existing scan-based design for testability (DFT) mechanisms so it requires simpler testing circuitry [2].

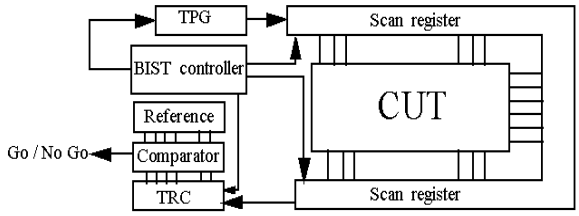

Fig. 6: Basic Block diagram of the serial BIST.

The built-in logic block observer register (BILBO) is one of the structures designed specifically for test-per-clock BIST schemes. It combines the function of a register, shift register (as a conventional scan flip-flop in Fig. 7), ALFSR (as a TPG), and MISR (as a TRC) built around one set of flip-flops. The circuit flip-flops are converted into BIST flip-flops. Two BIST flip-flops have been developed as part of my investigation into the BIST. Fig. 8 and Fig. 9 show two BIST flip-flops, referred to as BILBO cell1 and BILBO cell2, respectively.

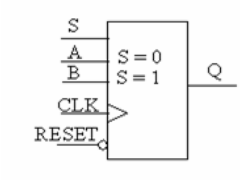

Fig. 7: Conventional scan flip-flop.

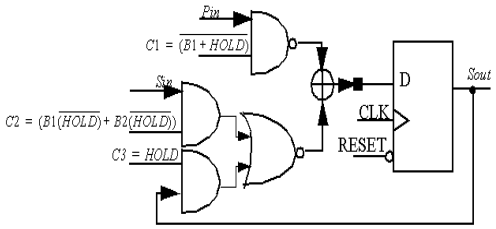

Fig. 8: BILBO cell1.

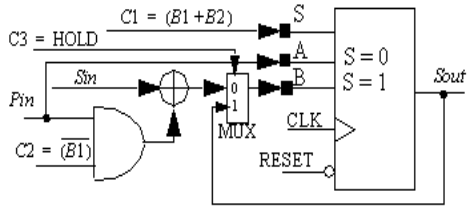

Fig. 9: BILBO cell2.

These cells can be reconfigured using control signals B1, B2, and HOLD to perform the normal, test pattern generation, test response compaction, scan, and hold modes. These control signals are generated from the BIST controller. The hold mode is used to freeze the resulting signature when a test session is finished. All cells are connected into a single scan chain in the scan mode either to provide access to the signature at the end of the test session or to seed the TPG and the MISR compactor (initial seed). These cells are considered the foundation of the user-defined data registers in IEEE-1149.1 standard architecture that will be discussed later.

Assigned codes are chosen for B1, B2, and HOLD. They are based on the assumption that the normal mode and the scan mode operations of all BILBO registers are simultaneously performed and a BILBO register operates either in the TPG mode or the TRC mode for a particular test session. The binary values assigned to B1 and B2 need to have the same values in the normal mode (B1 = B2 = 0) and in the scan mode (B1 = B2 = 1). The binary values assigned to B1 and B2 need to have different values in the TPG mode and in the TRC mode. B1 = 1 and B2 = 0 in the TPG mode, and B1 = 0 and B2 = 1 in the compaction mode. In the hold mode, HOLD signal is high where, in the BILBO cell1, both B1 and B2 are X (don't care) and, in the BILBO cell2, either one of B1 and B2, or both of them need to be high [7, 24]. Every BILBO register needs one distributed decoder to decode the control signals (B1, B2, and HOLD) into other appropriate control signals for the BILBO cells. BILBO cell1 and BILBO cell2 were compared in [2].

There are four main testing types in the BIST architecture. They are exhaustive testing [2, 21-22, 33], pseudo-exhaustive testing (PET) [2, 27-30], pseudo-random testing [21-22, 33], and deterministic testing [2, 34-36]. In this paper, the PET is selected to study. It achieves many of the benefits of exhaustive testing but usually requires far fewer test patterns. The time required for the PET depends on the sizes of the output cones shown in Fig. 10. The PET reduces the testing time to a feasible workable value whilst retaining many of the advantages of exhaustive testing.

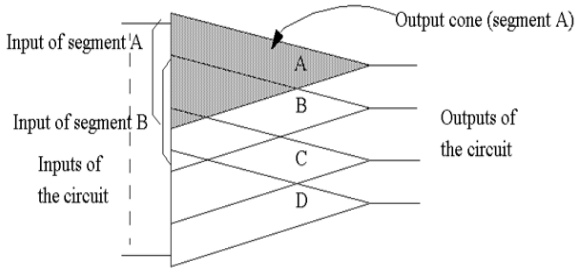

Fig. 10: Segmentation approach in the PET.

The circuit can be characterized as an (*n, m, k*) circuit. The circuit is segmented into *m* output cones, and each output cone is exhaustively tested, where *n* is number of circuit inputs. The PET does not require a separate fault coverage analysis as in the case of the pseudorandom testing. It is useful when no output depends upon more than *k* inputs ($k < n$), where *k* is small enough that a test length of 2^k is feasible. The test ensures detection of all detectable combinational faults within individual output cones of the CUT without the need for fault simulation in the case of the single-test pattern [8-10]. In addition, the test ensures detection of all detectable faults within individual output cones of the circuit without the need for fault simulation in the case of the two-test pattern [28]. If *k* is too large, it is necessary to segment the circuit such that some of the segment inputs and outputs are not primary inputs and outputs. Techniques for circuit segmentation have been discussed in [27, 29]. New output cones are generated through the insertion of segmentation cells.

In this paper, the steps of the BIST process are described through the simple structure shown in Fig. 11. This structure has three combinational blocks C1, C2, and C3. C1 is fed from the BIST BSR (Boundary Scan Register) input cells and feeds the BILBO register G2, C2 is fed from the BILBO register G2 and feeds the BILBO register G1. C3 is fed from the BILBO register G1 and feeds the BIST BSR output cells. This structure will be tested in two test sessions.C1, and C3 are tested in the first test session. In order to test C1, the BIST

BSR input cells are configured as a TPG and G2 is configured as a TRC. To test C3, G1 is configured as a TPG and the BIST BSR output cells are configured as a TRC. C2 is tested in the second test session with G2 configured as a TPG and G1 configured as a TRC. In Fig. 11, the control signal BIST_Inst_enable is active high when the BIST instructions are active. The BIST_Inst_enable is responsible to cause the cut in the case of the PET as shown in Fig. 11.

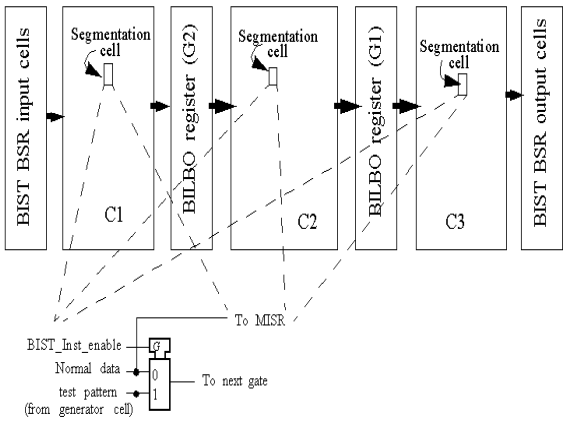

Fig. 11: Block diagram of the system (core) logic in the PET BIST.

The insertion of segmentation cells in the PET and the BILBO register adds significant area overhead and degrades circuit performance. In addition, the insertion of segmentation cells in the RTL may affect the system timing due to the unequal sequential depth in the BIST mode. The synchronous sequential circuit at the RTL can be viewed as combinational circuit interconnected with storage elements. The storage elements are assumed to be edge-triggered D-type flip-flops, and the sequential circuit representation in the graph model is an acyclic graph. For example, if the BIST capabilities of G1 and G2 in Fig. 11 are removed, both G1 and G2 become normal registers (parallel-in/ parallel-out). C1, C2, and C3 can be considered as one combinational block, C, called combinational equivalence. Now, the input of C is fed from the BIST BSR input cells and the output of C feeds the BIST BSR output cells. In the BIST mode, BIST BSR input cells are configured as a TPG and BIST BSR output cells are configured as an MISR and C is tested in one test session. The test pattern, applied to the combinational equivalence, is held for *d* clock cycles, where *d* is the maximum sequential depth of the circuit. The approach to pseudo-exhaustively test the combinational circuits can also be applied to this type of the sequential circuit without timing violation and less hardware overhead. In this paper, the BIST design solution to support these types of sequential circuits, based on the BIST boundary scan architecture (IEEE-1149.1) is presented.

3. DESIGN OF BIST BOUNDARY SCAN ARCHITECTURE

This section relates to a BIST method for an IEEE-1149.1 boundary scan circuit. The boundary scan register is designed to work in BIST mode (BSR-BIST). The instruction register supports new instructions to BIST operations. The TAP controller (TAPC) is designed to control the BIST process.

3.1 Design of the BIST-BSR

This section presents BIST-BSR input/output cells. The Update (UPD) flip-flops in the BSR input cells have been configured to operate as a TPG in the BIST mode. The Capture (CAP) flip-flops in the BSR input and output cells have been configured to operate as an MISR in the BIST mode. Fig. 12 presents a BIST BSR input cell for a primary input pin. The input pin of the chip is connected to *Pin*, and *Cin* feeds the core (system) logic input. Comparing the circuit with that of the standard cell, two new input signals (*Cin_p*, *BIST_mode_I*) have been added plus one multiplexer at the input of the UPD flip-flop. When *BIST_mode_I* is set high, the input cells are configured as part of a TPG in the BIST mode. Signal *Cin_p* is fed from signal *Cin* of the previous input cell and signal *Cin* feeds signal *Cin_p* of the next input cell. This cell requires 28.5 *gate equivalent* (GE), where a standard BSR input cell requires 26.5 GE so this cell requires more 2 GE to operate in the BIST.

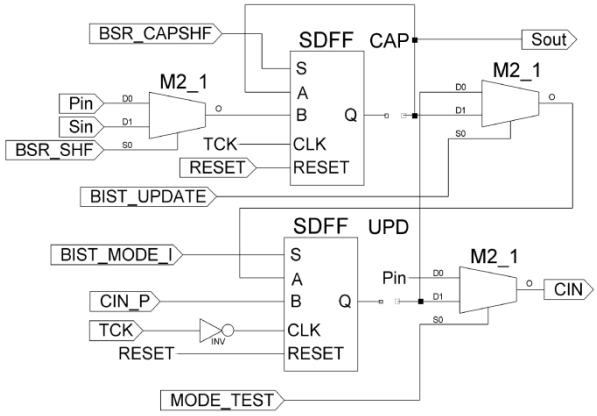

Fig. 12: Schematic diagram of the BIST-BSR primary input cell.

Fig. 13 presents a BIST-BSR output cell for a primary output pin. The output pin of the chip is connected from *Pout*, and *Cout* is fed from the core (system) logic output. Comparing the circuit with the standard cell, one new input signal has been added plus a multiplexer and an XOR gate at the input of the CAP flip-flop. When *BIST_mode_O* is set high, the cell is configured as part of an MISR in the BIST mode. This cell requires 32.25 GE, where a standard BSR input cell requires 26.5 GE so this cell requires 5.75 GE to operate in the BIST.

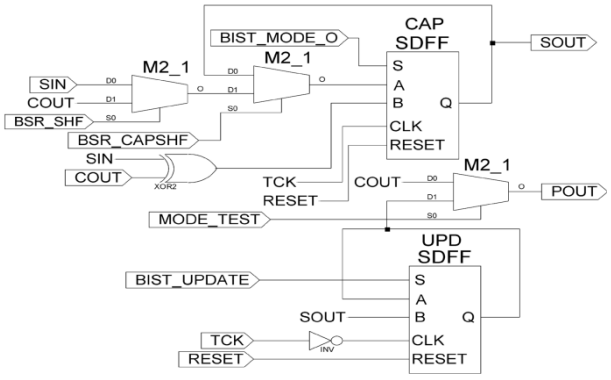

Fig. 13. Schematic diagram of the BIST-BSR primary output cell.

3.2 Design of the TAPC

The TAPC consists of two main blocks a finite state machine (FSM) and TAPC decoder, shown in Fig. 14. The FSM is driven by the signals TCK, TMS, and TRST* (optionally). The state transition diagram of the FSM, shown in Fig. 15, has sixteen states. It has a single control input, TMS, to determine the transitions between states which only occur on a rising edge of TCK.

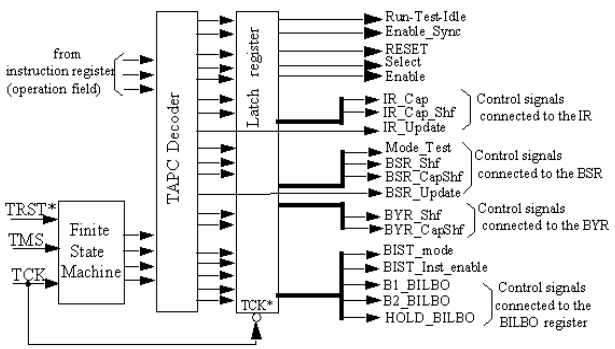

Fig. 14: Block diagram of the TAPC.

The inputs of the TAPC decoder are driven by four signals from the FSM and other three signals from the operation field of the instruction register (IR). All outputs of this decoder are latched into *D*-type flip-flops on the falling edge of TCK, except for two control signals, *IR_Update* and *BSR_Update*. These two control signals are latched on the rising edge of TCK because the UPD register in the BSR and shadow register in the IR are updated on the falling edge of TCK. The control bus, generated by TAPC and its latch flip-flops, is classified according to the selected register connected between TDI and TDO as shown in Fig. 14.

The control signals, which feed the *IR*, *BSR*, and bypass register (*BYR*), have been discussed in [2, 18]. Control signals *B1_BILBO*, *B2_BILBO*, and *HOLD_BILBO*, connected to B1, B2, and HOLD of the distributed decoder of the BILBO register (based on BILBO cell1), and control signals *BIST_mode*, and *BIST_Inst_enable* will be discussed here. The control signal *Enable_Sync* is used to switch between the chip clock and the clock of the boundary scan circuitry (TCK).

In this paper, eight instructions are used. Four of them are public instructions defined by the standard. The others are user-defined instructions which support the BIST operation. To support eight instructions, the operation field of the instruction word should be three bit in length. The number of test data registers used in this paper is three registers, (BSR, BYR, user-defined register). The address field length should be at least two bits. A four-bit address field is used to make the total instruction register length equal seven bit, shown in Fig. 16. The list of these instructions with their binary codes is presented in Table 1. The three most significant bits of each instruction define the operation field, and the remaining four bits form the address field.

Table 1. Instruction codes

Instruction	**Operation Code**	**Operation Field**	**Address Field**
Sample/Preload	000XX00	000	XX00
EXTEST	001XX00	001	XX00
BIST-BSR	010XX00	010	XX00
BIST-BILBO (BFT)	011XX01	011	XX01
BIST-BILBO (BST)	100XX01	100	XX01
SYNC	$101P_3P_2P_1P_0$	101	$P_3P_2P_1P_0$
INTEST	110XX00	110	XX00
BYPASS	111XX11	111	XX11

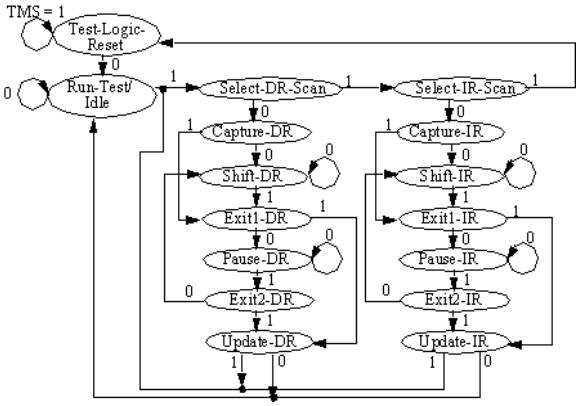

Fig. 15: State transition diagram of the FSM.

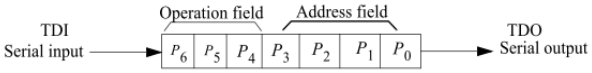

Fig. 16: Structure of the instruction word.

The IEEE-1149.1 standard defines nine public test instructions, three of them (SAMPLE/PRELOAD, EXTEST, and BYPASS) are mandatory [15]. The designer can also choose to incorporate additional user-defined instructions. Generally, the instructions are divided into two main types, non-invasive operational instructions and pin-permission operational instructions. In non-invasive instructions, the TAPC and TAP pins operate asynchronously and independently of the system (core) logic. This allows the boundary scan logic to be used without disturbing the normal operation of a chip, board or system. In pin-permission instructions, the I/O pins are disconnected from the system (core) logic and the BSR controls the I/O core logic.

The following section briefly describes the function of each of the defined public instructions. Binary instruction codes are given where they are mandated by the standard for some instructions. It is recommended, the all-zero and all-one operational code (op-code) map to non-invasive instructions. Each instruction may have more than one instruction code. Instructions that have no specified instruction code may be assigned an arbitrary code by the designer, provided that the code is unique to that instruction. The purpose of each instruction follows.

***BYPASS instruction*:** The BYPASS instruction is a non-invasive operational instruction. It places the single-bit BYR between TDI and TDO. Its purpose is to produce a short one-bit shift path through a chip, and for this chip to operate normally. This instruction must be selected by the binary instruction code consisting of logic 1 in each instruction register stage. The standard also states that all unused instruction codes, not declared to be user-defined, must also decode to BYPASS instruction [15].

***SAMPLE/PRELOAD instruction*:** The SAMPLE/PRELOAD instruction is a non-invasive operational instruction and the standard does not specify an instruction bit pattern for it. It serves a dual purpose. Firstly, it may be used to sample the logic values that are present on the chip pins without causing any disruption to the normal operation of the system (core) logic. Secondly, it may be used to set up (preload) the contents of the BSR prior to the execution of a pin-permission instruction in order to establish the desired initial conditions on the chip I/O pins to eliminate board level conflicts. Board level conflicts can be controlled by assuring that the chip

outputs are held to known safe values by BSR output cells during the test.

EXTEST instruction: The EXTEST instruction is a pin-permission instruction, with the choice of instruction bit pattern is left to the designer. The EXTEST instruction connects the BSR between TDI and TDO. When this instruction is active, the logical connection between the chip input pins and the internal system logic of the chip is broken and all chip output pins are under control of the BSR cells. This allows the values placed on the chip output pins to be easily controllable via the TAP. This instruction allows test patterns to be applied to the off-chip system logic and board level interconnects. During the EXTEST instruction, the chip input pins can be sampled and the chip output pins can be controlled. At the CAPTURE-DR state, all chip inputs are captured. Shifting the BSR during the SHIFT-DR state allows the user to read out captured input logic values and to set up new output logic values that will become effective upon passing through the UP-DATE-DR state. The EXTEST instruction is the workhorse of BS testing.

***INTEST instruction*:** The instruction INTEST is a pin-permission operational instruction and the standard does not specify an instruction bit pattern for it [15]. It is similar to that of the EXTEST instruction, but it is intended to test the internal system (core) logic of a device. It connects the BSR between TDI and TDO. It puts the system (core) logic inputs under control of the BSR input cells. The BSR output cells, connected from core logic outputs, sample the logic values produced by the core logic at the CAPTURE-DR state. Thus, at the UPDATE-DR state, a test pattern can be applied to the core logic inputs, and at the CAPTURE-DR state, the results of that test can be sampled. These results can be shifted out and a new test pattern can be shifted in. While this is happening, the logic values, driven to the output pins, are controlled by the BSR output cells so that known safe values are held during the test. Board level conflicts can be controlled by assuring that the chip outputs are held to benign values by the BSR. The INTEST instruction differs from EXTEST instruction in that the system logic may be clocked for a desired number of clock cycles which may be derived from TCK, before the test results are captured.

SAMPLE/PRELOAD instruction, EXTEST instruction, INTEST instruction, and BYPASS instruction are public instructions, discussed in [2, 15]; the others are user-defined instructions which support the BIST process. The purpose of these instructions follows.

***BIST-BSR instruction*:** The instruction BIST-BSR is a pin-permission operational instruction. It places the BSR between TDI and TDO, and puts the system (core) logic inputs under the control of the UPD flip-flops of BIST BSR input cells. During shifting, the signature generated in the BIST BSR cells after finishing BIST process can be shifted out and a new initial seed can be shifted in. While this is happening, the logic values, driven to the output pins, are controlled by the BIST BSR output cells so that known safe output values are held during the BIST process. At the UPDATE-DR state, an initial seed can be applied to the UPD flip-flops. At the RUN-TEST/IDLE state, the control signal *BIST_mode* goes high on the falling edge of TCK and this signal feeds the *BIST_mode_O* input of the BIST BSR output cells. Control signal *BIST_mode* is shifted a half clock cycle to feed the *BIST_mode_I* input of the BIST BSR input cells. The shifted version of the control signal *BIST_mode* goes high on the next rising edge of TCK. The FSM stays in the RUN-TEST/ IDLE state for a specific number of clock cycles. When the FSM leaves this state, the signature generated in the BIST BSR cells needs to be shifted out at the SHIFT_DR state. The FSM goes through the CAPTURE_DR state before going to the SHIFT_DR state. The BSR does nothing in the CAPTURE_DR state to avoid the corruption of the signature (control signal *BSR_CapShf* remains low at the CAPTURE_DR state).

***BIST-BILBO, first test session, referred to as BFT instruction*:** The BFT instruction places the user-defined register (BILBO register) between TDI and TDO. During shifting, the signature can be shifted out and a new initial seed can be shifted in. At the UPDATE-DR state and the CAPTURE_DR state, the register is in the hold state.

The BILBO registers have been divided into two groups for two test sessions. The control signals *B1_BILBO* and *B2_BILBO* feed B1 and B2 of the BILBO register of the first group, respectively. The control signals *B1_BILBO* and *B2_BILBO* are permuted to feed B2 and B1 of the BILBO register of the second group, respectively.

In the BIST mode, the first group operates as a TPG in the first test session and as an MISR in the second test session, and the second group operates as an MISR in the first test session and as a TPG in the second test session. These groups are configured as a single shift register in the shift mode. The storage elements are assumed to be the edge-triggered D-type flip-flops.

In the BFT instruction, and at the RUN-TEST/IDLE state, the control signal *B1_BILBO* goes high and *B2_BILBO* goes low on the falling edge of TCK. The FSM stays in the RUN-TEST/IDLE state for a specific number of clock cycles. When the FSM leaves this state, the signature generated needs to be shifted out at the SHIFT_DR state.

***BIST-BILBO, second test session, referred to as BST instruction*:** The BST instruction places the user-defined register (BILBO register) between TDI and TDO. During shifting, the signature can be shifted out and a new initial seed can be shifted in. At the UPDATE-DR state and the CAPTURE_DR state, the register is in the hold state. In this instruction, and at the RUN-TEST/IDLE state, control signal *B1_BILBO* goes low and *B2_BILBO* goes high on the falling edge of TCK. The FSM stays in the RUN-TEST/IDLE state for a specific number of clock cycles. When the FSM leaves this state, the signature generated needs to be shifted out at the SHIFT_DR state.

***SYNC (Synchronizing instruction)*:** This instruction will be discussed in section 4.

The truth table of the TAPC decoder outputs is shown in Table 2 and Table 3. The sequence of the control signals in each column of the table is as follows: RESET, Enable, Select, Enable_Sync, IR_Cap, IR_Cap_Shf, IR_Update, BSR_CapShf, BSR_Shf, BSR_Update, BYP_Shf, BYP_CapShf, Mode_Test, BIST_mode, BIST_Inst_enable, Hold_BILBO, B1_BILBO, B2_BILBO, Run-Test-Idle.

3.3. Test sequence of the BIST boundary scan

In this section, the steps of the BIST process are described through the simple structure shown in Fig. 11. This structure has three combinational blocks C1, C2, and C3. C1 is fed from the BIST BSR input cells and feeds the BILBO register G2, C2 is fed from the BILBO register G2 and feeds the BILBO register G1. C3 is fed from the BILBO register G1 and feeds the BIST BSR output cells. This structure will be

tested in two test sessions. C1 and C3 are tested in the first test session. In order to test C1, the BIST BSR input cells are configured as a TPG and G2 is configured as an MISR. To test C3, G1 is configured as a TPG and the BIST BSR output cells are configured as an MISR. C2 is tested in the second test session with G2 configured as a TPG and G1 configured as an MISR.

The control signal *BIST_Inst_enable* is active high when the instructions BIST-BSR, BFT, and BST are active. The control signal *BIST_Inst_enable* is responsible to cause the cut in the case of the PET as shown in Fig. 11. The TAPC controls the sequence of these two test sessions. The following steps will summarize the BIST process:

In the first test session:

1. Initialize the FSM of the TAPC to the TEST-LOGIC-RESET state.

2. Load the IR with the PRELOAD instruction. This connects the BSR between TDI and TDO, but does not grant pin-permission.

3. Shift the initial seed into the BIST BSR cells. While this is happening, the initial seed, driven to the BIST BSR primary output cells, controls the output pins so that known safe output values are held during the BIST process.

4. Load the IR with the BFT instruction. This connects G1 and G2 between TDI and TDO, and grants pin-permission.

5. Shift the initial seeds into G1 and G2 at the SHIFT_DR state.

6. Go to the RUN-TEST/IDLE state. In this state, control signal *B1_BILBO* goes high and *B2_BILBO* goes low to configure G2 as an MISR and G1 as a PETPG. The control signal *BIST_mode* goes high on the falling edge of TCK and this signal feeds the *BIST_mode_O* input of the BIST BSR output cells. Control signal *BIST_mode* is shifted a half clock cycle to feed the *BIST_mode_I* input of the BIST BSR input cells on the following rising edge of TCK. Stay in this state for the required number of clock cycles.

7. When the FSM leaves this state, the signature generated in G2 needs to be shifted out at the SHIFT_DR state.

Table 2. Truth table of the TAPC decoder outputs for the BS process

State assignment of FSM		**Instruction Sample/Preload**	**Instruction EXTEST and INTEST**	**Instruction BYPASS**
0000	Exit2-DR	110000000000000X000	110000000000100X000	110000000000000X000
0001	Exit1-DR	110000000000000X000	110000000000100X000	110000000000000X000
0010	Shift-DR	100000011000000X000	100000011000100X000	100000000011000X000
0011	Pause-DR	110000000000000X000	110000000000100X000	110000000000000X000
0100	Select-IR-Scan	111000000000000X000	111000000000100X000	111000000000000X000
0101	Update-DR	110000000100000X000	110000000100100X000	110000000000000X000
0110	Capture-DR	110000010000000X000	110000010000100X000	110000000001000X000
0111	Select-DR-Scan	110000000000000X000	110000000000100X000	110000000000000X000
1000	Exit2-IR	111000000000000X000	111000000000100X000	111000000000000X000
1001	Exit1-IR	111000000000000X000	111000000000100X000	111000000000000X000
1010	Shift-IR	101001000000000X000	101001000000100X000	101001000000000X000
1011	Pause-IR	111000000000000X000	111000000000100X000	111000000000000X000
1100	Run-Test/Idle	111000000000000X001	111000000000100X001	111000000000000X001
1101	Update-IR	111000100000000X000	111000100000100X000	111000100000000X000
1110	Capture-IR	111011000000000X000	111011000000100X000	111011000000000X000
1111	Test-Logic-Reset	011000000000000X000	011000000000000X000	011000000000000X000

Table 3. Truth table of the TAPC decoder outputs for the BIST process

State assignment of FSM		Instruction BIST-BSR	Instruction BIST-BILBO (BFT)	Instruction BIST-BILBO (BST)	Instruction SYNC
0000	Exit2-DR	110000000000101X000	1100000000001011110	1100000000001011110	110100000000100X000
0001	Exit1-DR	110000000000101X000	1100000000001011110	1100000000001011110	110100000000100X000
0010	Shift-DR	100000011000101X000	1000000000001010110	1000000000001010110	100100000000100X000
0011	Pause-DR	110000000000101X000	1100000000001011110	1100000000001011110	110100000000100X000
0100	Select-IR-Scan	111000000000101X000	1110000000001011110	1110000000001011110	111100000000100X000
0101	Update-DR	110000000100101X000	1100000000001011110	1100000000001011110	110100000000100X000
0110	Capture-DR	110000000000101X000	1100000000001011110	1100000000001011110	110100000000100X000
0111	Select-DR-Scan	110000000000101X000	1100000000001011110	1100000000001011110	110100000000100X000
1000	Exit2-IR	111000000000101X000	1110000000001011110	1110000000001011110	111100000000100X000
1001	Exit1-IR	111000000000101X000	1110000000001011110	1110000000001011110	111100000000100X000
1010	Shift-IR	101001000000101X000	1010010000001011110	1010010000001011110	101101000000100X000
1011	Pause-IR	111000000000101X000	1110000000001011110	1110000000001011110	111100000000100X000
1100	Run-Test/Idle	111000011000111X001	1110000110001110101	1110000110001110011	111100000000100X001
1101	Update-IR	111000100000101X000	1110001000001011110	1110001000001011110	111100100000100X000
1110	Capture-IR	111011000000101X000	1110110000001011110	1110110000001011110	111111000000100X000
1111	Test-Logic-Reset	011000000000000X000	0110000000000000X000	0110000000000000X000	0110000000000000X000

8. Load the IR with the BIST-BSR instruction. This connects the BSR between TDI and TDO, and grants pin-permission.

9. The signature, generated in the BIST BSR output cells, needs to be shifted out at the SHIFT_DR state. While this is happening, the logic values, driven to the output pins, are controlled by the BIST BSR primary output cells so that known safe values are held during the BIST process.

In the second test session:

10. Load the IR with BST instruction. This connects G1 and G2 between TDI and TDO, and grants pin-permission.

11. Shift the initial seeds into G1 and G2 at the SHIFT_DR state.

12. Go to the RUN-TEST/IDLE state. In this state, control signal *B1_BILBO* goes low and *B2_BILBO* goes high to configure G1 as an MISR and G2 as a PETPG. Stay in this state for the required number of clock cycles.

13. When the FSM leaves this state, the signature generated in G1 needs to be shifted out at the SHIFT_DR state.

14. Go to the TEST-LOGIC-RESET state and halt the test.

4. SINGLE-CLOCK TEST APPROACH

In the single-clock test approach, all boundary scan cells are clocked by *TCK* (Test Clock). All registers in the system (core) logic are clocked by *Chip_CK* in the normal mode and by *TCK* in the test mode. *Chip_CK* and *TCK* are asynchronous clocks. Fig. 17 illustrates the clock distribution.

A programmable control unit, PCU, shown in Fig. 18, is designed to hold the TPG and the MISR each test pattern for at least *d* clock cycles, where *d* is the maximum sequential depth. It enables the architecture to convert the problem of sequential circuits into one combinational block. The control of the PCU is based on *SYNC* instruction (user-defined instruction). It sets control signal *Enable_Sync* to high and switches between the chip clock and the TCK. The address field of the *SYNC* instruction is loaded to the programmable counter, *PC*, at inputs, *P0, P1, P2, P3* when both control signals *Enable_Sync*, and *Run-Test-Idle* go high. (Control signal *Run-Test-Idle* is active high in the Run-Test/Idle state) These inputs represent the equivalent binary form of *d*. For example, for $d = 3$, the binary form is 0010 ($d - 1 = 2$), and *P3, P2, P1, P0* is 1101. The carry signal, generated from *PC*, will be high for a clock cycle every *d* clock cycles. The *PC* is enabled when control signal *BIST_mode* goes high. The control signals *BIST_mode_O* and *BIST_mode_I* , which feed the BIST BSR output and input cells, respectively, and *HOLD_BILBO_in* which feeds the BILBO registers, are shown in Fig. 18 and Fig. 19 (d =16). The control signals *DR_CapShf* and *DR_Shf*, which feeds all BIST BSR cells, are also shown in Fig. 18. These control signals will enable the BIST BSR input/output cells and the BILBO registers to hold for *d-1* clock cycles in the BIST mode. Control signal *Enable_Sync* control multiplexer *M1*. For normal operation and before loading the *SYNC* instruction, the Chip_CK feeds the system logic. When the *SYNC* instruction is loaded in the test mode, *Enable_Sync* goes high on the falling edge of *TCK*. The *SyEnable flag* is set on the following rising edge of TCK, the *DivRun flag* is set on the following falling edge of TCK. Multiplexer *M1* is switched to select its second channel (*TCK*), which feeds the system logic.

Using programmable control unit (PCU) for register transfer level, the following steps will summarize the BIST process:

1. Initialize the FSM of the TAPC to the Test-Logic-Reset state and load the IR (Instruction Register) with the PRELOAD instruction.

2. Shift the initial seed into the BIST BSR cells.

3. Load the IR with the SYNC instruction. Set *d* to 16 (*P3 P2 P1 P0* = 0000). Now, all registers in the system logic are clocked by TCK.

4. Load the IR with the BIST-BSR instruction. The IR may be loaded with the BIST-BSR instruction again to enable the propagation of the response of the initial seed through the normal registers to the inputs of the BIST BSR output cells.

5. Go to the Run-Test/Idle state. In this state, the control signals *BIST_mode_O*, *BIST_mode_I*, *HOLD_BILBO_in*, *DR_CapShf*, and *DR_Shf*, generated from PCU, will be as shown in Fig. 19. Stay in this state for the required number of clock cycles.

6. When the FSM leaves this state, the signature, generated in BIST BSR cells, needs to be shifted out.

7. Go to the Test-Logic-Reset state and halt the test.

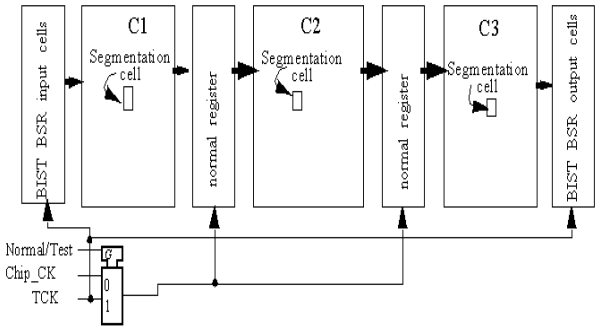

Fig. 17 Clock distribution.

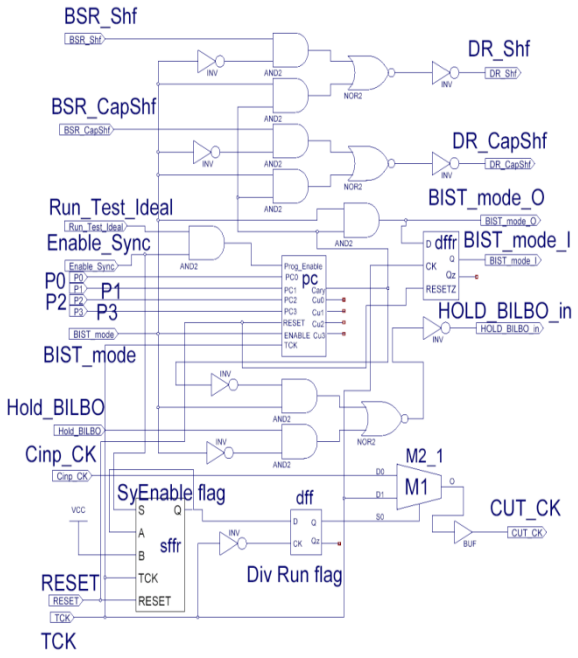

Fig. 18 Schematic diagram of the programmable control unit (PCU).

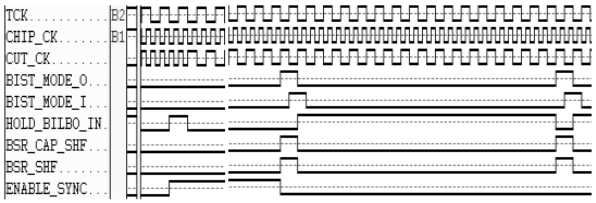

Fig. 19 Timing diagram using the programmable control unit.

5. DESIGN AND IMPLEMENTATION OF A TESTABLE 16-BIT PARALLEL PIPELINED MULTIPLIER

This section presents a complete design of a boundary scan testable circuit. The system (core) logic implements 16-bit parallel pipelined multiplier. The complete testable 16-bit parallel pipelined multiplier with BIST boundary scan architecture is presented.

5.1 Design and Implementation of 16-Bit Parallel Pipelined Multiplier

The basic form of multiplication consists of forming the product of two positive binary numbers. The valid result for the multiplication operation will be available after sixteen clocks. Fig. 20 illustrates the block diagram of 16-bit parallel pipelined multiplier. The block *16-bit* is used as one clock shift (register). The system (core) logic implements 16-bit parallel pipelined multiplier. M-16x1 consists of two 16-bit register cell, one 16x1 multiplier cell, and 16-bit adder cell. All these cells are connected altogether and implemented on Spartan Xilinx (X2C100). The timing simulation of the complete design, shown in Fig. 21, illustrates the proper operation under applying the input stimulus. The hexadecimal input 1D1C multiplies the hexadecimal input 009C that equals 0011BD10 after sixteen clocks.

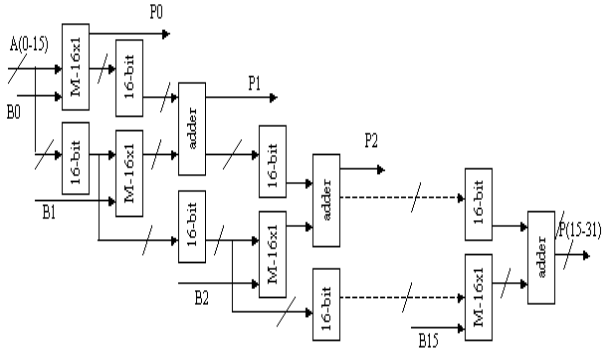

Fig. 20: Block diagram of 16-bit parallel pipelined multiplier.

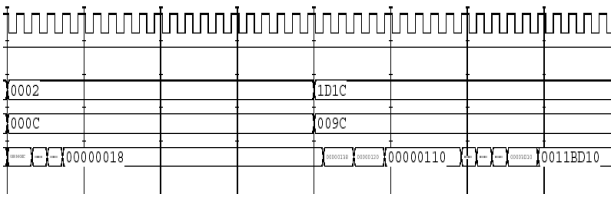

Fig. 21: Timing diagram of the 16-bit parallel pipelined multiplier.

The report generated due to this implementation is given as follows:

Minimum period: 36.418ns (Maximum Frequency: 27.459MHz)	
Number of Slices:	758 out of 1,200 63%
Number of Slices containing unrelated logic:	0 out of 758 0%
Number of Slice Flip Flops:	79 out of 2,400 19%
Number of 4 input LUTs:	1,240 out of 2,400 51%
Number of bonded IOBs:	65 out of 140 46%
Number of GCLKs:	1 out of 4 25%
Number of GCLKIOBs:	1 out of 4 25%
Total equivalent gate count for design: 11,272	

5.2 Design and Implementation of a Testable 16-Bit Parallel Pipelined Multiplier

The 16-bit parallel pipelined multiplier is chosen to verify test circuitry. We will start with the discussion of the test circuitry and then the testable 16-bit parallel pipelined multiplier. The test circuitry added to the multiplier here is based on the design of the single-clock test approach presented in section 4. The test pattern, applied to the combinational equivalence, is held for 16 clock cycles, where *d* is the maximum sequential depth of the circuit. The method in section 3 to pseudo-exhaustively test the combinational circuits can also be applied to this type of the sequential circuit without timing violation and less hardware overhead. The design solution to support these types of circuits was presented. All boundary scan cells are clocked by *TCK*. All registers in the system (core) logic are clocked by *Chip_CK* in the normal mode and by *TCK* in the test mode. *Chip_CK* and *TCK* are asynchronous clocks. Fig. 17 illustrates the clock distribution. A programmable control unit, PCU, shown in Fig. 18, is to hold the TPG and the MISR each test pattern for at least 16 clock cycles, where 16 is the maximum sequential depth. It enables the architecture to convert the problem of sequential circuits into one combinational block. All cells of test circuitry are carefully connected. The BIST BSR input cells and BIST BSR output cells are connected in series format to achieve the input and output pins of the testable multiplier, the input pins are connected to a constant value (high or low) for test reason point of view to overcome the problem of floating pins.

All cells of the multiplier and test circuitry are connected altogether and implemented on FPGA chip Xilinx (X2C100), shown in Fig. 22. The timing simulation of the complete design is presented to verify proper operation. It is clear from the simulation as shown in Fig. 23 that the generated design netlist operated correctly under applying the input stimulus. It represents the timing simulation of the testable 16-bit parallel pipelined multiplier. Enable_Sync control signal goes high when the Sync instruction is applied; the clock applied to CUT is switched to TCK from chip clock. The PCU hold the *DR_Shf*, *DR_CapShf*, *BIST_Mode_O*, and *BIST_Mode_I* sixteen clocks for proper timing.

The report generated due to implementation is given as follows:

Minimum period: 119.976 ns (Maximum Frequency: 8.335MHz)	
Number of Slices:	1,012 out of 1,200 84%
Number of Slices containing Unrelated logic:	0 out of 1,012 0%
Number of Slice Flip Flops:	654 out of 2,400 27%
Number of 4 input LUTs:	1,689 out of 2,400 70%
Number of bonded IOBs:	109 out of 140 77%
Number of GCLKs:	2 out of 4 50%
Number of GCLKIOBs:	2 out of 4 50%
Total equivalent gate count for design: 15,372	

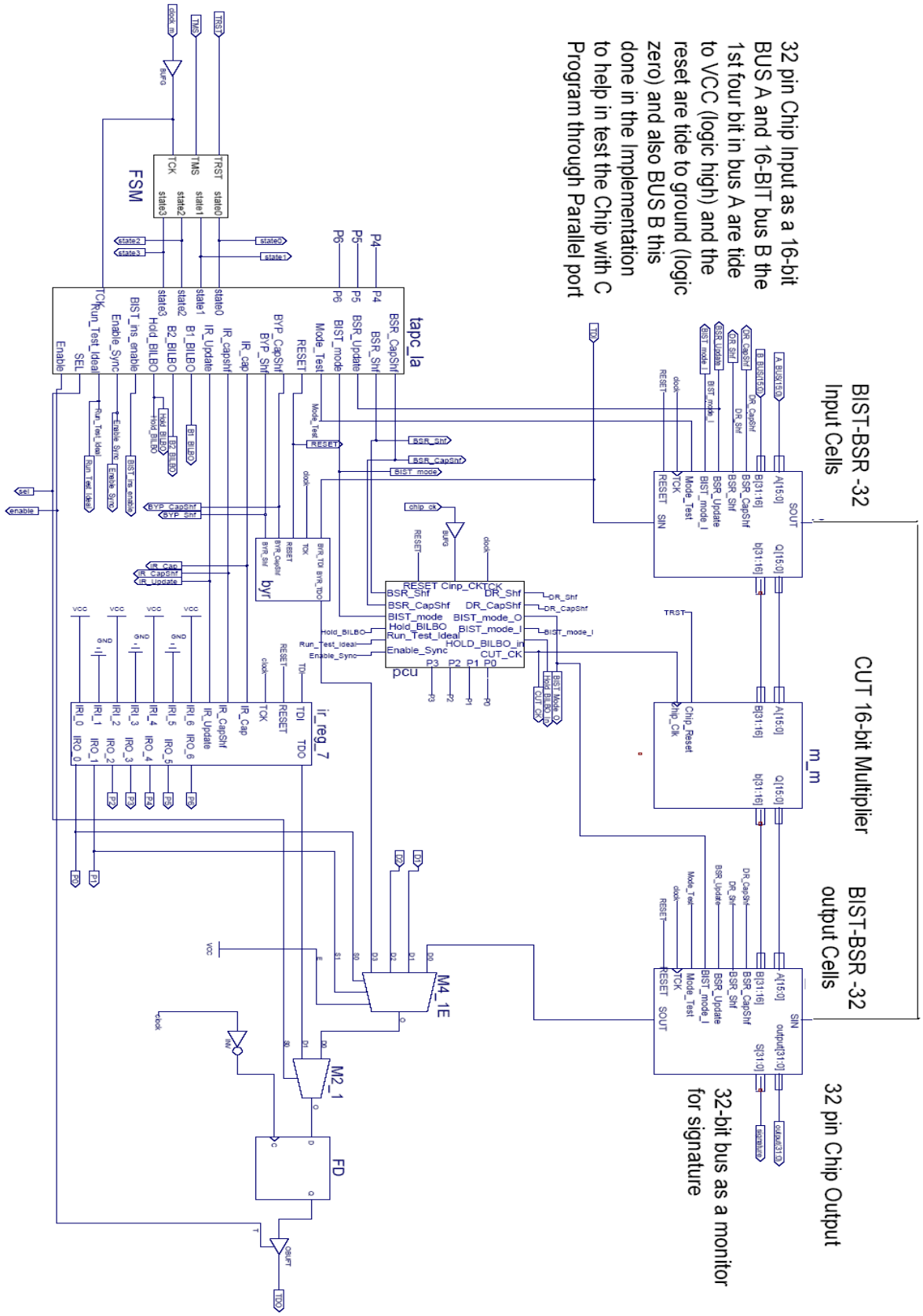

Fig. 22: Schematic diagram of the testable 16-bit parallel pipelined multiplier.

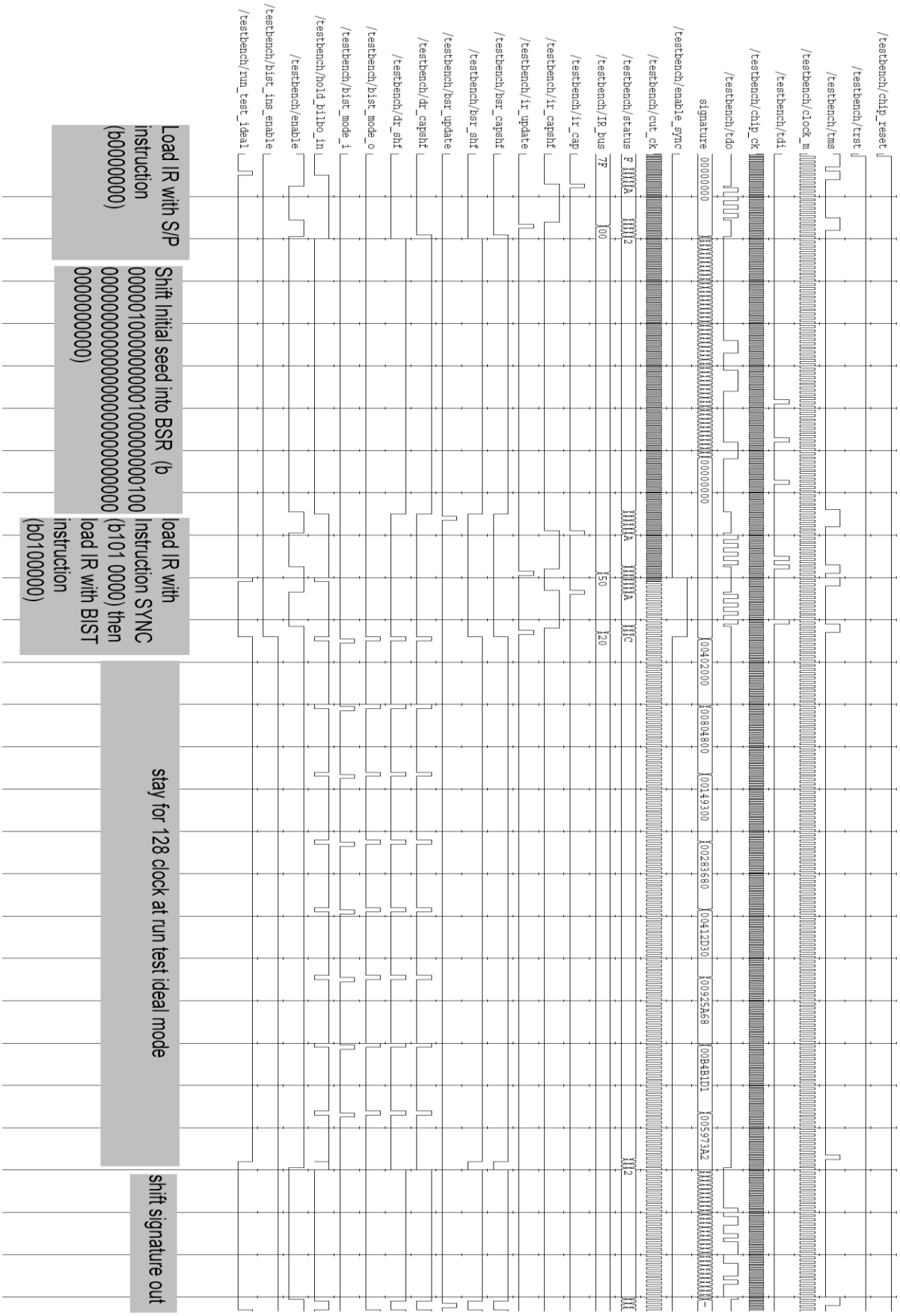

Fig. 23: Timing diagram of the testable 16-bit parallel pipelined multiplier.

The report generated due to implementation shows that the 16-bit parallel pipelined multiplier runs at about 27.459MHz clock speed and it occupies about 11272 gate equivalent (GE) on Spartan X2C100 device. The hardware implementation results show that the testable 16-bit parallel pipelined multiplier runs at about 8.335MHz clock speed and it occupies about 15372 GE on Spartan X2C100 device. The hardware overhead due to the adding test circuitry is 4100 GE, and the maximum clock speed is reduced in the case of testable design.

The hardware overhead of the both BIST BSR I/O is 46 GE and the total number of the BIST BSR implemented in the testable design is 64 cells. The hardware overhead due to the BIST BSR is 2944 GE. The overhead of the BIST BSR input/output cells is high compared to the hardware required by the rest of the boundary scan circuitry. By placing the BIST BSR input/output cells in the area between the I/O pad and core logic, the area overhead in the core area can be reduced. So, the hardware overhead due to the rest boundary scan test circuitry, which is nearly constant for every design, is 1156 GE. This hardware overhead is low with respect to the complete testable design.

6. EXPERIMENT RESULTS

In this section, the evaluation of the design performance on the FPGA chip as a real time application system is presented. An experiment is established for that purpose. In this paper, the BIST boundary scan architecture for register transfer level (RTL) was presented with the 16-bit parallel pipelined multiplier. All designed cells are connected as shown in Fig. 22. The timing diagram represents the simulation as shown in Fig. 23. All design steps are processed for the implementation on the FPGA chip Spartan Xilinx chip X2C100. The download card has a FPGA chip Spartan X2C100 of capacity 100K gate, in addition to some other components that are required to monitor and facilitate the card operation. The card was mainly designed to be a complete computer system that can be interfaced to I/O peripherals like a monitor, the parallel port, a keyboard and a mouse. Fig. 24 illustrates the photo copy picture of the download card.

Fig. 24: Photo copy picture of the download card.

The timing simulation results are taken for the five pins (TRST, TMS, TCK, TDI, and Chip_CLK) at every transition clock to generate the test digital vector with the expected binary values of TDO and stored in a text file. The test digital vectors are applied to the parallel port, and then to Spartan Xilinx chip X2C100. Table 4 represents the input/output binary signals generated from the parallel port with C program. The signals are TRST (D0), TMS (D1), TCK (D2), TDI (D3), Chip_CLK (D4) and TDO (S5).

Table 4. Input/output binary signals generated.

TRST	0111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
TMS	111111111111000011111110000000000000000000000000000000000000000000000000011111110000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000001111111111110000000000000000000000000000000000000000000011111110000111111100000000000000000000000000000000000000000000000000000111111100000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000111100000000000000000000000000000001 100000000000011111110000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000011110000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
TCK	001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110 011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100 110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001 100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011 001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110 011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100 110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011
TDI	000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000011110000000000000000000000000000000000000001111000000000000000000000000000000000000000111100000000000000000000 000000000000000000000000000000000111100000000011110000111100000000000000000000000000000000000000000000000000000111100000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
Chip_CLK	010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010 101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101 010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010 101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101 010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010 101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101 010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101
TDO	zzzzzzzzzzzzzzzzzzzzzzzzzzzzz**000111100001111000011110000111100**zzzzzzzzzzzzzzzzz**0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000** **000000000111111111111100000000011111111111100000000000000000000000000000000000000000000000000000011111110000000000000000000000000000**zzzzzzzzz**0111111111111000000000000000000000000000** **000000**zzzzzzzzzzzzzzzz**00111100001111000011110000111100**zzzzzzzzzzzzzzzzzzzzzzzz**1111110000111100001111000011110**zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz**00000000111100000000011110000111100001111000011110000000001111000000000111111111111100001111000000000011** **11**zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz

7. CONCLUSIONS

The incorporation of BIST capabilities into the boundary scan architecture, defined in the IEEE-1149.1 standard was presented. This paper presented new boundary scan architecture for BIST. This structure has the following capabilities:

- The UPD flip-flops of the BIST BSR input cells are configured to form a TPG in the BIST mode. The CAP flip-flops of the BISR BSR input and output cells are configured to form an MISR in the BIST mode. This configuration supports BIST for either the cascaded or non-cascaded BSR input and output cells.
- The TAPC controls the BIST process; in addition to the standard boundary scan process. Three instructions for BIST process are presented to support the BIST operation for the BSR and the BILBOs (user defined registers).

The structure, given in the paper, proposes that the chip operates in two test sessions. In the case where the chip needs more than two test sessions, it is easy to design other user-defined instructions to deal with other test sessions.

For the RTL, the insertion of segmentation cells in the case of the PET may affect system timing due to the unequal sequential depth in the BIST mode and add significant area overhead due to the delay flip-flop. Also, transferring every flip-flop into BIST flip-flop adds area overhead and degrades circuit performance in the normal mode of the chip. To compensate these problems, one new design solution with boundary scan architecture was proposed. The incorporation of BIST capabilities into the boundary scan architecture with this solution was presented and the TAPC controls the BIST process. The first design uses single-test clock. The programmable control unit enables the TPG to hold each test pattern for at least *d* clock cycles, where *d* is the maximum sequential depth. The structure here proposes that the chip operates in single test sessions.

In this paper, the FPGA implementation of the BIST boundary scan architecture for 16-bit parallel pipelined multiplier was presented. The schematic based design methods were used to design the complete testable design. The hardware implementation results showed that the hardware overhead due to the adding test circuitry is 4100 gate equivalent, and the maximum clock speed is reduced in the case of the testable design. The hardware overhead of the BIST BSR input/output cells is high compared to the hardware required by the rest of the boundary scan test circuitry. By placing the BIST BSR input/output cells in the area between the I/O pad and core logic, the area overhead in the core area can be reduced. So, the hardware overhead due to the rest of the boundary scan test circuitry, which is constant in every design, is 1156 GE. This hardware overhead is low comparing to the complete testable design. Designing testable circuits and interfacing it with the computer were presented to evaluate the design as a real time application. The hardware experiment results were compared with the simulation results to verify the design performance.

8. REFERENCES

[1] Mohamed H. El-Mahlawy, *Automatic Measurement of Digital Circuits*, M.Sc. thesis, Military Technical College, Egypt, 1995.

[2] M. H. El-Mahlawy, "*Pseudo-Exhaustive Built-In Self-Test for Boundary Scan*", Ph.D. thesis, Kent University, U.K., 2000.

[3] M. El Said Gohniemy, S. Fadel Bahgat, Mohamed H. El-Mahlawy, and E. E. M. Zouelfoukkar, "A Novel Microcomputer Based Digital Automatic Testing Equipment using Signature Analysis." IEEE International conference on Industrial Applications in Power Systems, Computer Science and Telecommunications, pp. 140-144, Bari, Italy, May 13-16, 1996.

[4] Mihalis Psarakis, Dimitris Gizopoulos, Ernesto Sanchez and Matteo S. Reorda, "Microprocessor Software-Based Self-Testing", *IEEE Design & Test of Computers*, Vol. 27, No. 3, pp. 4-19, May/June, 2010.

[5] Sherif I. Morsy, Mohamed H. El-Mahlawy, Gouda I. Mohamed, "Design for Testability Technique for Microcontroller", 8th *International Conference of the Electrical Engineering*, Egypt, 29-31 May 2012.

[6] Sherif I. Morsy, Mohamed H. El-Mahlawy, and Gouda I. Mohamed, "Hybrid based Self-Test Solution for Embedded System on Chip", *International Journal of Computer Applications*, Volume 84, No. 12, pp. 7-14, December 2013.

[7] Mohamed H. El-Mahlawy, "Signature Multi-Mode Hardware-Based Self-Test Architecture for Digital Integrated Circuits," *IEEE International Conference on Electronics, Circuits, & Systems*, pp. 437-441, 6-9 Dec. 2015.

[8] Mohamed H. El-Mahlawy, and Winston Waller, "An efficient algorithm to design convolved LFSR/SR." 17th National Radio Science Conference, Egypt, pp. C23 (1-10), Feb. 2000.

[9] Mohamed H. El-Mahlawy, and Winston Waller, "New Test Pattern Generators for the BIST Pseudo-Exhaustive Testing based on Coding Theory Principles", Communications on Applied Electronics (CAE) Journal, Volume 4, No. 8, pp. 29-44, April 2016.

[10] Mohamed H. El-Mahlawy, and Winston Waller, "A New Single Test Pattern Generator for Pseudoexhaustive Testing.", *11th International Conference on Aerospace Sciences & Aviation Technology* (*ASAT-11*), Egypt, pp. 989-1002, 15-17 May 2005.

[11] M. H. El-Mahlawy, M. S. Hamed, M. H. Abd-El-Zeem, and I. Yossef, "FPGA Implementation of the BIST IP For SRAM Chips", Proceedings of the 6th International Conference of the Electrical Engineering (ICEENG-6), Military Technical College, Cairo, Egypt, 27-29 May 2008.

[12] Mahmoud S. Ragab, Mohamed H. El-Mahlawy and Emad A. El-Samahy, "Efficient Microcontroller System to Test an SRAM Chip Using Signature Analysis," 13th International Computer Engineering Conference (ICENCO), pp. 388-392, Dec 27-28, 2017.

[13] Emad H. Khalil, M. H. El-Mahlawy, Fawzy Ibrahim and M. H. Abdel-Azeem, "Design for Testability of Circuits and Systems; An overview", 5th International Conference of the Electrical Engineering, Egypt, May 16-18, 2006.

[14] Mohamed H. El-Mahlawy, Ehab A. El-Sehely, Al-Emam S. Ragab, and Sherif Anas, "Design and Implementation

of a New Built-In Self-Test Boundary Scan Architecture." *IEEE 15th International conference on Microelectronics*, pp. 27-31, 9-11 Dec. 2003.

[15] Kenneth P. Parker, "The Boundary-Scan Handbook," 3rd Edition, Norwel: Kluwer Academic, 2003.

[16] Sherif Anas, "*In-Circuit Testing for Electronic Board*", M.Sc. thesis, Military Technical College, Egypt, 2006.

[17] S. S. Wasouf, H. N. Ahmed, Mohamed H. El-Mahlawy, M. M. Mokhtar, "A Proposed Boundary Scan Testing Module for Automatic Testing of Digital Integrated Circuits", Proceedings of the 13th International Conference on Aerospace Sciences & Aviation Technology (ASAT-13), Cairo, Egypt, May 26-38, 2009.

[18] Mohamed H. El-Mahlawy, "Architecture for BIST Boundary Scan." 10th International Conference on Aerospace Sciences & Aviation Technology (ASAT-10), Egypt, 13-15 May, 2003.

[19] T. T. S. C. IEEE C. Society, "IEEE Standard Test Access Port and Boundary Scan Architecture – IEEE Std. 1149.1,", 2001.

[20] T. T. S. C. IEEE C. Society, "IEEE Standard for Boundary-Scan Testing of Advanced Digital Networks," in *IEEE Computer Society*, 2003.

[21] Paul H. Bardell, Willian H. McAnney, Jacob Savir, *Built-In test for VLSI: pseudorandom techniques*, John Wiley and Sons, 1987.

[22] S. W. Golomb, *Shift Register Sequences*, Laguna Hills, CA: Aegean Park Press, 1982.

[23] Mohamed. H. El-Mahlawy and A. Seddik, "Design and Implementation of New Automatic Testing System for Digital Circuits Based on the Signature Analysis", Proceedings of the 12th *International Conference on Aerospace Sciences & Aviation Technology* (*ASAT-12*), Egypt, 29-31 May 2007.

[24] Mohamed. H. El-Mahlawy, A. Abd El-Wahab, and A.S. Ragab, "FPGA Implementation of the Portable Automatic Testing System for Digital Circuits", Proceedings of the 6th *International Conference of the Electrical Engineering* (*ICEENG-6*), Egypt, 27-29 May, 2008.

[25] M. H. El-Mahlawy, "A Novel Testing Method for Monostable Multivibrators", 5th *International Conference of the Electrical Engineering* (*ICEENG-5*), Egypt, 16-18 May 2006.

[26] Mohamed H. El-Mahlawy, "Signature-Based Self-Test Approach for Single-Shot Circuits on the Circuit Board Level," The Fourth International Japan-Egypt Conference on Electronics, Communications and Computers (JEC-ECC 2016), pp. 38-42, May 31 - June 2, 2016.

[27] Mohamed H. El-Mahlawy, Winston Waller, "An efficient algorithm to partition the combinational circuits for pseudoexhaustive testing." 17th National Radio Science Conference, Egypt, pp. C24 (1-11), Feb. 22-24, 2000.

[28] Mohamed H. El-Mahlawy, Emad H. Khalil, Fawzy Ibrahim, and Mohamed. H. Abd El-Azeem, "Two-Test Pattern Capabilities of the LFSR/SR Generator in Pseudo-Exhaustive Testing based on Coding Theory Principles", European Journal of Scientific Research, Volume 140 – No. 2, pp.161-177, July 2016.

[29] Mohamed H. El-Mahlawy, and Winston Waller, "New Algorithm to Segment Combinational Circuits in Pseudo-Exhaustive Testing", European Journal of Scientific Research, Volume 140 – No. 1, pp.40-58, June 2016.

[30] Chih-Ang Chen and Sandeep K. Gupta, "BIST Test Pattern for Two-Pattern Testing- Theory and Design Algorithms*", IEEE Transactions on Computers*, Vol. 45, NO. 3, March 1996.

[31] L. -T. Wang and E. J. McCluskey, "Concurrent built-in logic block observer (CBILBO)," *IEEE International Symposium on Circuits and Systems*, Vol. 3, pp. 1054-1057, Oct. 1986.

[32] Wu E., "PEST: A tool for implementing pseudoexhaustive self-test," AT&T Technical Journal, Vol. 70, No.1, pp. 87-100, Jan. 1991.

[33] Thomas W. Williams, Wilfried Daehn, Matthias gruetzner, and Cordt W. Starke, "Bounds and analysis of aliasing errors in linear feedback shift register", *IEEE transactions on computer-aided design*, Vol. 7, NO. 1, pp 75-83, Jan. 1988.

[34] Angela Krstic, and Kwang-Ting (Tim) Cheng, "*Delay Fault Testing for VLSI Circuits*", Kluwer Academic Publishers, 1998.

[35] Mukund Sivaraman, and Andrzej J. Strojwas, "*A unified Approach for Timing Verification and delay Fault Testing"*, Kluwer Academic Publishers, 1998.

[36] Miron Abramovici, Melvin A. Breuer, Arthur D. Friedman, "*Digital systems testing and testable design*," IEEE press, Inc., New York, 1994.