IJEE 0973-7383
International Journal of Electronics Engineering (ISSN: 0973-7383)
Volume 11 • Issue 1 pp. 54-61 Jan 2019-June 2019 www.csjournals.com

An Efficient Fault Detection of FPGA Using Low Transition - Random Test Pattern Generation With Light Weighted Code and EX-OR GATE

Dr. D. Mahesh Kumar
Associate Professor in Electronics, PSG College of Arts & Science, Coimbatore

Abstract: In this paper, a new technique for localization of fault detection and diagnosis in the interconnects and logic blocks of an arbitrary design implemented on a Field-Programmable Gate Array (FPGA) using BIST is presented. This technique can uniquely identify any single bridging, open or stuck-at fault in the inter connect as well as any single functional fault; a fault resulting a change in the truth table of a function, in the logic blocks. The test pattern generator and output response analyzer are configured by existing CLBs in FPGAs. Thus, no extra area overhead is needed for the proposed BIST structure. The scheme also rests on partitioning of rows and columns of the memory array by employing low cost test logic. It is designed to meet requirements of at-speed test thus enabling detection of timing defects. Experimental results confirm high diagnostic accuracy of the proposed scheme and its time efficiency.

Keywords: Fault diagnosis, Built-in self-test (BIST), Configurable Logic Block (CLB), Field-Programmable Gate Array(FPGA), Testing.

I. INTRODUCTION

Field-Programmable Gate Arrays (FPGAs)are 2-D arrays of Configurable Logic Blocks (CLBs) and programmable switch matrices, surrounded by programmable input/output blocks on the periphery. FPGAs are widely used in many applications such as networking, storage systems, communication, and adaptive computing, due to their reprogrammability, flexibility, and reduced time-to-market. The reprogrammability of FPGAs results in faster design anddebug cycle compared to Application-Specific Integrated Circuits(ASICs). However, once the design is finalized and fixed, the programmability becomes useless and costly if further infield customization and reprogrammability are not required. In order to reduce the manufacturing costs associated with FPGAs, application-specific FPGAs have been introduced in the FPGA industry which restricts the use of the FPGA device for only one application (design). Xilinx's Easypath solution isan example [1]. The cost reduction is mainly due to using devicesthat may contain defects in the areas not used by the particular application. This, in turn increases the manufacturing yield compared to the traditional scenario in which any defective device is thrown away.

During system operation, application-dependent test and diagnosis are very crucial in online self-repair schemes for fault tolerant applications [2]. In these applications, the existence of faults in the system is first identified and faulty resources are precisely diagnosed afterwards. Then, the design is remapped to avoid faulty resources. Because test and diagnosis procedures are performed during system operation (online), the number of test vectors and configurations must be minimized. Note that the test time is dominated by loading test configurations rather than applying test vectors. Compared to application-independent test and diagnosis, application-dependent test and diagnosis provides faster test and diagnosis time while achieving a higher diagnosis resolution over a more comprehensive fault list. This is because application-dependent test and diagnosis focus only on the FPGA resourcesused for that particular design, rather than all FPGAresources.

For interconnect diagnosis, the configuration ofused logic blocks is modified, and the configuration of the interconnectsremains unchanged. Any single fault (open, stuck-at,or bridging fault) in the interconnects can be uniquely identifiedin a small number of test configurations. For logic diagnosis, aBuilt-in Self Diagnosis (BISD) method is presented in which theconfiguration of used logic blocks remains unchanged while theconfigurations of the interconnect resources and unused logicblocks are modified. Any single functional fault, inclusive of allstuck-at faults, in logic blocks is precisely diagnosed in only onetest configuration.

The use of memory cores in SOCdesigns is rising quickly. As memory cores are dominating thesilicon area of typical SOC designs, and the density of memorycircuits is normally higher than logic circuits, the chip yield ismainly determined by the memory yield. To improve the chipyield, whether by process enhancement or design improvement,diagnosis of the memory cores after testing is necessary. Embeddedmemory testing is normally done by Built-in Self-Test(BIST) [3], [4]. A BIST scheme that also collects and exportsthe diagnostic data for subsequent online or offline analysis hasbeen called a Built-in Self-Diagnosis (BISD) scheme [5], [6].

Just as test data compression for logic circuits, memory testdata compression has also received attention recently. In [7], thebit-maps for large memories are compressed by using fail patterns. Another work considers the compression of the output responseof the BIST circuit [8]–[10]. The method is similar to signatureanalysis in logic BIST. The BIST circuit may export thetest information, called fault-syndrome, to test

for failure analysis.The size of fault-syndrome affects total test cost directly.

However, except diagnosis data compression, using a redundancyrepair approach to enhance memory yield is the other importantissue in recent years. It has become imperative to deploy effective means fortesting and diagnosing non-volatile memory failures. A functionalmodel employed for these memories remains similarto that of RAMs with relevant fault types such as stuck-atsand bridges being tackled through functional test algorithms[11]. Also, all addressing malfunctions are covered by memorycell stuck-at fault tests as there are no writes in the missionmode. Typically, the basic test reads successive memory cells,and processes output responses by performing a polynomialdivision to compute a cyclic redundancy code (signature).The same procedure can be used to detect certain classesof dynamic faults provided memory cells that are designed withadditional DFT features [12].

A novelBIST design with comprehensive on-the-fly exhaustive redundancysearch and analysis method is presented in [13], whichallows on-chip optimal redundancy allocation without havingto construct the complete fail bitmap. It however has high hardwareoverhead for a reasonably big number of spare (redundant)elements.The three types of fail patterns has been found such as: faulty words, faulty rows, and faulty columns. The faultyrow/column is the continuous faults on the same row/column. Different fail patterns exhibit different syndrome characteristics.The built-in syndrome compressor is designed to efficientlycompress the fault syndromes. Our approach reducesthe amount of data that need to be transmitted from the chipunder test.

Moreover,the proposed method does not increase the test time for thefault-free memories. It results in a muchshorter diagnosis time than the conventional BISD schemes.Simulation results for memories under various fault pattern distributionsshow that in most cases the data can be compressed toless than that of its original size. Furthermore, based on fail patternidentification technique, the faulty row/column can be replacedby redundancy row/column. Therefore, the complexityof RA algorithms can be reduced. An acceptable RA algorithmfor BIST implementation should consider not only the repair efficiencybut also the hardware overhead of the BISR circuit.

II.BASIC BIST ARCHITECTURE

A representative architecture of the BIST circuitry as it might be incorporated into the CUT is illustrated in the Fig 1. This BIST architecture includes two essential functions as well as two additional functions that are necessary to facilitate execution of the self-testing feature while in the system.

Fig. 1 Basic BIST

Architecture

The two essential functions include the Test Pattern Generator (TPG) and Output Response Analyzer (ORA). While the TPG produces a sequence of patterns for testing the CUT, the ORA compacts the output responses of the CUT into some type of Pass/Fail indication. The other two functions needed for system-level use of the BIST include the test controller (or BIST controller) and the input isolation circuitry. Aside from the normal system I/O pins, the incorporation of BIST may also require additional I/O pins for activating the BIST sequence (the BIST Start control signal), reporting the results of the BIST (the Pass/Fail indication), and an optional indication (BIST Done) that the BIST sequence is complete and that the BIST results are valid and can be read to determine the fault-free/faulty status of the CUT.

III. LOW POWER – BIST

Low Power - BIST (LP-BIST) shown in Figure 2(a) & 2(b) is another example of test-per-clock approach for the concurrent type and in that the comparator monitors the normal operation data. If it is the same as the pattern in the LFSR, the test clock is ticked [25]. The response is fed to the MISR for the compression and LFSR advances one clock cycle. If there is no match for a long time, the LFSR generated test clock is ticked once automatically to advance one test cycle. At the same time, the system clock is held for one cycle.

Fig. 2(a) EX-OR based Pattern on BIST Architecture

Fig. 2(b) Inverter based Pattern on BIST Architecture

The proposed BIST for test-per-scan is implemented to meet the following criteria:

- Can reduce switching activity in CUT'S during BIST and reduced power consumption.
- Does not degrade CUT performance.
- Can be implemented with little area overhead.

A BIST TPG that can achieve high fault coverage and also reduce switching activity during BIST is proposed for single scan chain designs in, which augments are the LP-TPG with the inverter and EX-OR based pattern generation with gated clock. It is shown that the inverter based pattern generation can achieve very high fault coverage with low hardware overhead. The Low Transition - Random Test Pattern Generator (LT-RTPG) proposed, generates correlated test patterns that can reduce transitions at state inputs during scan shift operations. The serial fixing LT-RTPG can also generate test patterns that cause less switching activity during BIST.

Figure 2(a) and 2(b) show the block diagram of BIST TPG. Here the pattern generated from LFSR is given to XOR-LP-TPG and inverter-LP-TPG. Mode select is mainly used to select the faults detected from LP-TPG i.e. the undetected faults from LP-TPG and is given to scan chain for shifting and storing of patterns and then to CUT, as more switching activity cause damage to the circuit. Here the LP-TPG reduces switching activities in the circuit so that the number of transitions will be reduced and less power will be consumed.

The built in self-test pattern generator is comprised of two test pattern generators:

- Low Transition Random TPG.
- 3-weight Weighted Random BIST architecture.

A multiplexer is used to selectively drive the inputs to the scan chain. The multiplexer that drives the input of scan chain, selects a test pattern source between the LP-TPG using inverter and LP-TPG using inverter BIST [18]. In the first test session, test patterns generated by the LT-TPG are selected and scanned into the scan chain to detect easy-to-detect faults. In the second session, test patterns that are generated by the 3-weight weighted random built in self-test are selected to detect the faults that are mainly undetected after the first session [25]. An outline of the overall procedure is to design an optimized one.

EX-OR BIST TPG by the proposed method is described below

1. Apply a sequence of test patterns generated by the LP XOR-TPG to the circuit and drop all detected faults.
2. $i \leftarrow 0$,
3. Initialize the current test cube set, $C_i \leftarrow \varphi$, and generator $(C_i) \leftarrow \{X,X.....X\}$.$j \leftarrow 0$ and unmark all faults in the fault list.
4. If there are no faults in the fault list, then exit. If else select an unmarked fault f i that has the minimum test generation cost and generate a test cube Cj for the fault by the proposed ATPG.
5. Add the test cube C_j to the current test cube set, $C_i \leftarrow C_i \cup C_j$.$j \leftarrow j+1$ based on gated logic.
6. Update generator according to the definition. If the number of conflicting inputs is smaller than or equal to U max (a positive integer), then mark all faults detected by test cube and go to step (4).
7. $C_i \leftarrow C_i - C_j$. Update generator and generate patterns by using EX-OR based pattern generator. Run fault simulation to drop the faults that are detected by the generated LP-XOR-TPG patterns. $i \leftarrow i+1$ and go to step (3).

IV. FPGA FAULT DETECTION

The interconnect resources in FPGAs can be categorizedas inter-CLB and intra-CLB resources. Inter-CLB routingresources provide interconnections among CLBs. Inter-CLBresources include programmable switch blocks and wiringchannels connecting switch blocks and CLBs. Intra-CLB resourcesare located inside each CLB. Intra-CLB interconnectsinclude programmable multiplexers and wires inside CLBs.Diagnosing faults in inter-CLB routing resources is addressedin this section. For inter-CLB interconnect test and diagnosis,the configuration of routing resources remains unchanged whilethe configuration of logic resources is modified.

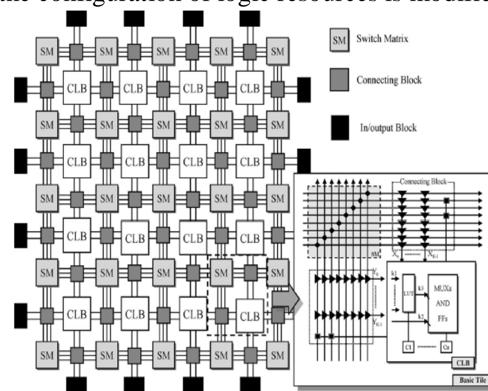

Fig. 3FPGA Architecture

Test and diagnosisof intra-CLB interconnects along with logic resources are also discussed. For this purpose, the configurationof used logic resources (inclusive of intra-CLB interconnects)is kept unchanged whereas the configuration of inter-CLBinterconnects as well as unused logic resources are changed.The separation between inter-CLB and intra-CLB is madebecause in contemporary FPGAs the programmable logic, resourcesare not limited to lookup tables (LUTs); other logic resourcessuch as carry generation/propagation logic and cascadechains are included in CLBs. For inter-CLB interconnect testand diagnosis, these logic elements, if used in the original configuration,will be bypassed.

A single-term function F is a logic function which has onlyone minterm or only one maxterm. In other

words, the truthtable of a single-term function consists of only one minterm orone maxterm. The input pattern corresponding to that minterm(or maxterm) of function F is called Activating Input (AIF).

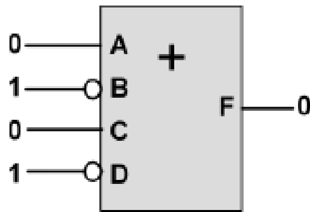

Fig. 4 Single-term Function with Activating Input Pattern

A single-term function can be viewed as an AND (OR) functionwith possible inversions at the inputs and/or output. For a single-term function, if the applied input vector is theactivating input, all sensitized faults are detected. An exampleis shown in Fig. 4, which has only one maxterm.

Since the activatinginput (0101) is applied, A/1 (A stuck-at 1 fault), B/0, C/1 and D/0 are detected.Moreover the bridging faults between A and B are also detected. It should be noted that if a bridging fault is sensitized, i.e.,two nets have opposite values, detection under various bridgingfault models, namely wired-OR, wired-AND, and dominant, isguaranteed. This is because the value of at least one of the signalsis modified and the condition of single-term function andactivating inputs guarantee the propagation of faulty value(s) tothe reachable primary output(s).

Detection of feedback bridging faults requires logic-levelsensitization and propagation of the fault. In addition to that,depending on the polarity of feedback path, which may resultin oscillation, some extra timing conditions must be satisfied.The use of single term function guaranty the logic-levelrequirements of such detection.

Single-term functions guarantee the detection of all sensitizedfaults. However, some mechanism is required to sensitize allfaults in the fault list.We implement single-term functions in allused LUTs in the design. By implementing different single-term functions in used logic blocks such that each fault in the fault list is sensitized in at least onetest configuration, all faults can bedetected.Since these test configurations target faults in inter-CLB interconnect,all additional logic resources in CLBs, if used, willbe bypassed. Hence, CLBs are configured as LUTs followedby flip-flops. The followingsubsections describe the proposed diagnosis procedures basedon various fault models.

(1) Diagnosis of Stuck-At Faults: A circuit with ‘n’ nets has 2nstuck-at faults. Based on the above assumption, in order touniquely identify any single stuck-at fault at least$\log_2 2n = 1 + \log_2 n$ test configurations are required.

(2) Diagnosis of Open Faults: An open fault on a net canbe detected by applying a sequence of stuck-at fault tests forthat net. Since an open fault can behave either as stuck-at-1 orstuck-at-0 faults, it is required to test for both stuck-at faults toguarantee the detection of open faults.If the logicbehavior of an open fault is equivalent to a stuck-at-1 (or stuck-at-0)fault, then the diagnosis procedure identifies the open fault as astuck-at-1 (or stuck-at-0) fault due to fault equivalency.

(3) Diagnosis of Bridging Faults: The bridging fault list for ‘n’, a circuit with nets contains n(n-1)/2 distinct pair-wisebridging faults. Hence, at least $\log_2 [n(n-1)]/2 = 2\log_2 n-1$ test configurations are required for single bridging faultdiagnosis.

The number of test configurations for bridging fault diagnosiscan be reduced if a smaller fault list is used. Note that a considerablenumber of n(n-1)/2 bridging faults (in a design with ‘n’ nets) cannot happen based on physical layout information usinginductive fault analysis (IFA) techniques [14]. If such faults areremoved from the fault list, the number of test configurationscan be reduced in a logarithmic scale.After faulty nets are diagnosed, if the exact failing interconnectresources (line segment, programmable switch, or multiplexer)within the faulty nets are required to be identified, highresolution interconnect diagnosis methods similar to those presentedin [15] can be exploited afterwards.

Consider an FPGA with N LUTs, such that each LUT has K inputs. The maximum number of nets for any designs to be mapped into this FPGA is N x (K+1). This means that one separate net is associated with every input and the output of eachLUT in the FPGA.

V. CONFIGURATION LOGIC BLOCK DETECTION

For logic block (including intra-CLB interconnects)testing and diagnosis, the configuration of the originallyused logic blocks is preserved while the configuration of interconnectsand unused logic blocks are changed to exhaustivelytest and diagnose all used logic blocks. This is in contrast to themethod presented in the previous section for interconnects inwhich the configurations of used CLBs are replaced by appropriatesingle-term functions.

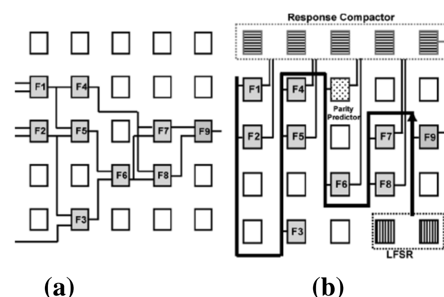

Fig. 5 Application-dependent Self-test Architecture for Logic Blocks (a) Original Configuration (b) BIST Configuration

The idea of application-dependent logic block testing is presentedin [16]. In this BIST scheme, each used logic block is exhaustively (or super-exhaustively, i.e., all possible transitions)tested while all these logic blocks are tested concurrently. The global interconnect is reprogrammed in such a way that the test signals are routed to each logic block. A Linear FeedbackShift Register (LFSR) or a binary counter for generating test vectors is connected to the inputs of all used logic blocks. The logic block outputs are observed through an internal response comparator (e.g., an XOR tree). The response comparator can be combined with a response (parity) predictor, as will be explained shortly, such that a unique pass/fail signal can be generated. The LFSR and the XOR tree are implemented in the available unused logic blocks. Since the LFSR or binary counter generates all possible patterns (2^n patterns for an-input logicblock) and the XOR tree propagates any single fault to its output,any single functional fault in the used logic blocks will be propagatedto the output of the XOR tree and will be detected. Functionalfaults are any faults that change the truth-table of an LUT,including stuck-at faults.

Fig. 5 shows an example of this scheme. In Fig. 5(a) theoriginal design, with used logic blocks F1 to F9 with originalinterconnections, is shown. In the BIST configuration, the originalinterconnections are modified such that LFSR outputs, implementedin unused blocks, are connected to the inputs of allused blocks F1-F9 in parallel. The outputs of used blocks alongwith the parity predictor block are connected to the responsecompactor, which is also implemented in the available unusedresources.The classical XOR tree does not provide any diagnosis capability.In order to improve the diagnostic resolution of this scheme, a combinational compactor based on error correctingschemes can be exploited instead of the XOR tree. If a compactorwith more than one output is used and logic blockoutputs are selectively connected to the compactor outputs(through the network of XOR gates), the failing pattern at theoutputs of the compactor can identify failing logic block(s). Test patterns generators (TPGs), such as LFSR, and OutputResponse Analyzers (ORAs) have been used in the contextof FPGA BIST [17], [18], [19], [20], [21]. However, the useof parity/checksum precomputation (which requires only oneLUT/block rather than a full XOR-tree) and response comparatorwhich uniquely codes the failing block(s), particularly in thecontext of application-dependent diagnosis, is novel.

VI. FAIL PATTERN IDENTIFICATIONIN MEMORY

In this work, the BIST and BISR design are based on failpatternidentification, and this section describes the fail-patternidentification scheme in detail. A defect in different parts of thememory may lead to different faults and/or fail patterns [22],[23], [24]. Fault identification is not trivial, and can be aidedby using the fail pattern information. We will describe the approachto distinguish the fail patterns during the test process.

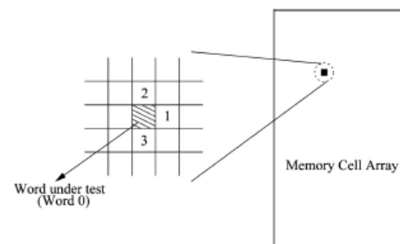

Fig. 6 Memory Cell Array Being Tested

Fig. 6 shows the memory cell array being tested. The shaded regionrepresents the Word UnderTest (WUT). If the WUT has adifferent output than the expected value when we read it, thenthe word fails and a fault is detected. In a typical BISD design,when a fault is detected the test process pauses, and the faultdata is either registered or shifted out before the test processresumes. However, whether it is shifted out immediately or registeredand then shifted out later, the cost (time complexity andATE capture memory size) can be high if there are many faults.We use a more advanced approach, i.e., identify the faulty rows,faulty columns, and faulty words simultaneously during the testprocess.

• **Faulty Row:** When theWUTis faulty, we test the next wordin the same row, i.e., Word 1 as shown in Fig. 1. If Word 1is also faulty, we continue to test the next word in the samerow until we reach a fault-free word or the end of the row.

• **Faulty Column:** Identification of a faulty column, assumingthe WUT has been tested faulty, consists of several condition-checking steps.

—Word 1 is tested fault free, so a faulty row can be excluded.

—The word above the WUT in the same column (i.e.,Word 2 as shown in Fig. 6) is tested faultfree; otherwisethe WUT has been covered by the previous faultycolumn test.

—Theword under theWUTin the same column (i.e.,Word3 as shown in Fig. 6) is tested faulty.We continue to testthe subsequent words in the same column until we reacha fault-free word or the end of the column.

• **Single Faulty Word:** When the WUT is faultybut not in afaulty row or column, i.e., Word 1, Word 2, and Word 3are all tested fault-free, we consider the WUT as a singlefaulty word.

This process does not increase the test time for afault-free memory, because the test algorithm is the same as in the original BIST design. If thememory is faulty, then there will be a slight time penalty for failpattern identification. However, compared with original BISDscheme, the size of memory diagnosis data to be exported andthe total test data diagnosis time can be reduced greatly. ConsideringBISR applications, the fail pattern identification approachcan replace the must-repair

phase and the time penalty can becompensated by RA time reduction.

Diagnosis Syndrome Format

The proposed fault syndromes for the three fail patterns, aswell as the original syndrome, are shown in Fig. 6. The originalsyndrome composes of three field-sessions, address, and word syndrome. The Session field records the Read operationthat detects the fault. The Address field stores the addressof the faulty word, so its length is equal to the length of anormalword address. TheWord Syndrome field stores the compressing word syndrome of the faulty word at the current state, whichrepresents the faulty cells in this word. The proposed syndromefor single faulty word has four fields-Syndrome ID, Session,Address, and CompressedWord Syndrome. The Syndrome IDsare used to distinguish the fail patterns: 00, 11, and 01 representthe single faulty word, row fault, and column fault, respectively.

The Faulty-Row syndrome is also composed of four fields.It does not include the Word Syndrome, but it needs to recordthe addresses of the first and last faulty words. Since the lastfaulty word has the same row address with the first faulty word,we only need to store the column address of the last faulty word(the End Column field). The Faulty-Column syndrome is similarto the Faulty-Row syndrome, except that it has the CompressedWord Syndrome field. Since all words are in the same column,only the address of the last faulty word in the column is recordedin the End Row field. Because the memory is word-oriented, theWord Syndrome is needed to locate the faulty bits (columns) inthe word. It is also compressed by the Huffman code. Note thatthe Faulty-Column syndrome may be longer than the originalsyndrome, but it actually represents multiple faulty words in thesame column, so it still has a high compression efficiency. Furthermore,to identify more number of fault types, ex. multirowfault or multicolumn fault, the number of Syndrome ID may beincreased. Different memory has different fault types. And differentfault types require different data format and compressionmethod. Moreover, the hardware cost may also increase to identifydifferent fault types. In this manuscript, we target on threetypical fault types: faulty-row, faulty-column, and faulty word.

If the memory is fault-free, the Sequencer will only run in theTest Execution states, i.e., BIST Idle, BIST Apply, and BISTDone. It will then look like a typical BIST design, and in thiscase the testing time does not increase.

VII. RESULTS AND DISCUSSION

Table 1shows the various parameters of the comparison between the existing method and the proposed method.The simulated outputs are shown in the fig. 7, 8 and 9 in thatthe simulation result for LP-BIST is shown in the fig. 7 and for the fig. 4 single term function without applying the BIST is shown in the fig. 8 same thing for the fault detection in the CLBs and also in the memory using BIST as shown in the fig. 9.

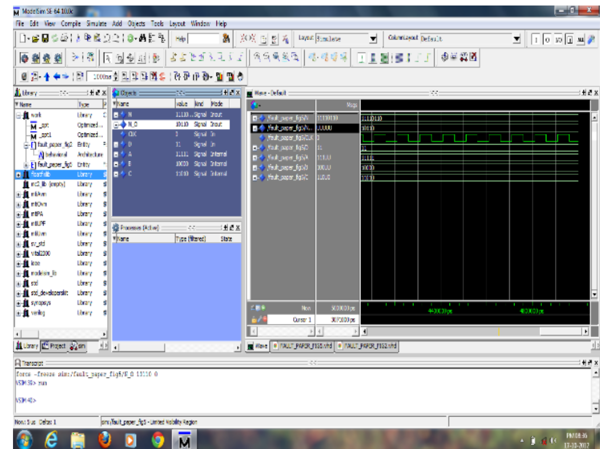

Fig. 7 Simulation Output for LP-TPG using EX-OR – Proposed Method

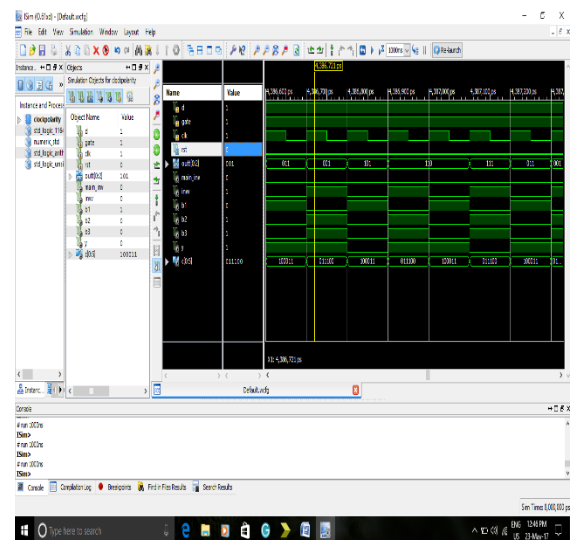

Fig. 8 Simulation for Single-term Function

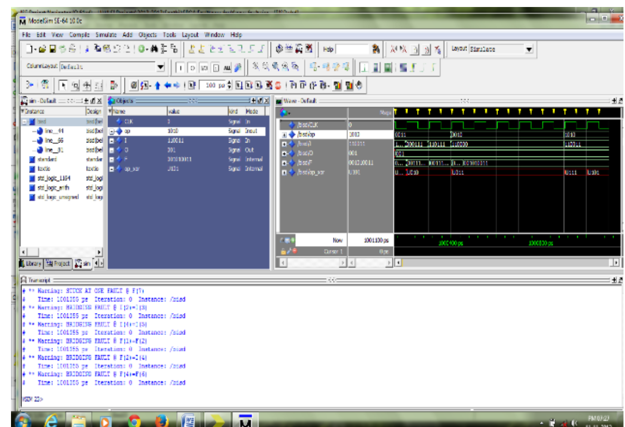

Fig.9 Simulation for Fault Detection in Memory

International Journal of Electronics Engineering (ISSN: 0973-7383)
Volume 11 • Issue 1 pp. 54-61 Jan 2019-June 2019 www.csjournals.com

Since it is very difficult to generate test patterns by a random pattern generator that causes minimal number of transitions while they are scanned into the scan chain and whose responses also cause a minimal number of transitions while they are scanned out of the scan chain.

Table 1 Comparison Result of Existing and Proposed Method

	Existing Method	Proposed Method	
PARAMETERS	**MAC**	**EX-OR**	**INV**
Number of 4 input LUTs	8 (out of 4896)	3 (out of 4896)	3 (out of 4896)
Number of occupied Slices	7 (out of 2,448)	4 (out of 2,448)	4 (out of 2,448)
BELS	18	7	5
Delay	3.061ns	2.289ns	2.289ns
Total Memory usage	238992	239568	239504
Total REAL time to Xst completion:	9.00 secs	7.00 secs	9.00 secs
Total CPU time to Xst completion	9.19 secs	6.71 secs	8.49 secs
Total REAL time to PAR completion	11 secs	9 secs	7 secs

VIII. CONCLUSION

This paper presents a low hardware overhead TPG for scan based BIST that can reduce switching activity in CUTs during BIST and also achieve very high fault coverage with an arguable length of the test sequence. The test patterns are produced by pseudorandom pattern generators such as LFSRs which requires very little hardware overhead. Achieving high fault coverage for CUTs that contain many Random Pattern Resistant Faults (RPRFs) only with (pseudo) random patterns generated by an LFSR often require unforeseeable long test sequences thereby resulting in prohibitively long test time. The proposed TPGs inverter LP-TPG and EX-OR LP-TPG reduces switching activities in the circuits, so that it reduces the number of transitions that occur at scan inputs during scan shifting by scanning in the test patterns where neighbouring bits are highly correlated and less power is consumed.

The proposed BIST comprises of two TPG inverter LP-TPG and TPG EX-OR LP-TPG. Test sequences generated by the inverter LT-TPG detect easy-to-detect faults. Faults that escape LT-RTPG test sequences are detected by test patterns generated by the EX-OR LP-TPG. The number of weight sets (generators) is minimized by guiding the proposed EX-OR LP-TPG with cost functions that reflect the number of conflicting inputs to be incurred by setting an input to a binary value. An algorithm to design the EX-OR LP-TPG that requires minimal hardware overhead and whose patterns cause a minimal number of transitions during scan shift cycles is presented. The proposed BIST structure does not require modification of mission logic, which can cause performance degradation. Experimental results for ISCAS'89/85 benchmark circuits demonstrate that the proposed BIST can significantly reduce switching activity during BIST while achieving maximum fault coverage for all benchmark circuits. Instead of LFSR we can use a dual speed LFSR where it will generate pseudo random test patterns and also it runs faster as compared to LFSR.

A new BIST approach for fault detection and diagnosisof FPGAs and memory has been proposed in this paper. Theproposed FPGA BIST structure has high fault coverage in themodelled interconnect and CLB faults, including short/open anddelay faults in wire channels, stuck on/off faults in PSs, andstuck-at-0/1 faults in LUTs. The test results for various FPGAs have shown that adequate performance in faultcoverage, test time, and area overhead can be achieved byusing the proposed BIST structure. The proposed FPGA BIST structurepossesses the ability to simultaneously detect and diagnosefaults on both interconnect resources and CLBs.For interconnect diagnosis,multiple faults (open, stuck-at, or bridging fault) canbe uniquely identified.

IX. REFERENCES

[1] Xilinx Inc., San Jose, CA, "Xilinx EasyPath Solutions," 2006. [Online].Available: www.xilinx.com
[2] W.-J. Huang and E. J. McCluskey, "Column-based precompiled configurationtechniques for FPGA fault tolerance," in Proc. IEEE Symp.Field-Program. Custom Comput. Mach., 2001, pp. 137–146.
[3] C.-T. Huang, J.-R. Huang, C.-F. Wu, C.-W. Wu, and T.-Y. Chang,"A programmable BIST core for embedded DRAM," IEEE Des. TestComput., vol. 16, no. 1, pp. 59–70, Jan.–Mar. 1999.
[4] L.-T. Wang, C.-W. Wu, and X. Wen, Design for Testability: VLSI TestPrinciples and Architectures. San Francisco, CA: Elsevier (MorganKaufmann), 2006.
[5] R. P. Treuer and V. K. Agarwal, "Built-in self-diagnosis for repairableembedded RAMs," IEEE Des. Test Comput., vol. 10, no. 2, pp. 24–33,Jun. 1993.
[6] C.-W. Wang, C.-F. Wu, J.-F. Li, C.-W. Wu, T. Teng, K. Chiu, andH.-P. Lin, "A built-in self-test and self-diagnosis scheme for embeddedSRAM," J. Electron. Test.: Theory Appl., vol. 18, no. 6, pp. 637–647,Dec. 2002.
[7] J.Vollrath, U. Lederer, and T. Hladschik, "Compressed bit fail maps formemory fail pattern classification," in Proc. IEEE Euro. TestWorkshop(ETW), 2000, pp. 125–130.
[8] V. N. Yarmolik, S. Hellebrand, and H. Wunderlich, "Self-adjustingoutput data compression: An efficient BIST technique for RAMs," inProc. Conf. Des., Autom., Test Euro. (DATE), 1998, pp. 173–179.
[9] S. Hellebrand, H. Wunderlich, A. Ivaniuk, Y. Klimets, and V. N.Yarmolik, "Error detecting refreshment for embedded DRAMs," inProc. IEEE VLSI Test Symp. (VTS), 1999, pp. 384–390.
[10] S. Hellebrand, H. Wunderlich, A. Ivaniuk, Y. Klimets, and V. N.Yarmolik, "Efficient online and offline testing of embedded DRAMs,"IEEE Trans. Comput., vol. 51, no. 7, pp. 801–809, Jul. 2002.

[11] A. K. Sharma, Semiconductor Memories: Technology, Testing and Reliability.New York: Wiley, 2002.
[12] Y.-H. Lee, Y.-G. Jan, J.-J. Shen, S.-W. Tzeng, M.-H. Chuang, and J.-Y.Lin, "A DFT architecture for a dynamic fault model of the embeddedmask ROM of SoC," in Proc. Int. Workshop Memory Technol. DesignTesting, 2005, pp. 78–82.
[13] T. Kawagoe, J. Ohtani, M. Niiro, T. Ooishi, M. Hamada, and H.Hidaka, "A built-in self-repair analyzer (CRESTA) for embeddedDRAMs," in Proc. Int. Test Conf. (ITC), 2000, pp. 567–574.
[14] S. R. Patil, D. B. Musle, "Implementation of BIST technology for fault detection and repair of the multiported memory using FPGA", International conference of Electronics, Communication and Aerospace Technology (ICECA), December 2017.
[15] M. B. Tahoori, "Diagnosis of open defects in FPGA interconnects," inProc. IEEE Int. Conf. Field-Program. Technol., 2002, pp. 328–331.
[16] M. B. Tahoori, "Application dependent testing of FPGAs," IEEE Trans.Very Large Scale Integr. (VLSI) Circuits, vol. 14, no. 9, pp. 1024–1033,Sep. 2006.
[17] M. Abramovici and C. Stroud, "BIST-based detection and diagnosisof multiple faults in FPGAs," in Proc. Int. Test Conf., 2000, pp.785–794.
[18] A. Doumar and H. Ito, "Detecting, diagnosing, and tolerating faults inSRAM-Based field programmable gate arrays, a survey," IEEE Trans.Very Large Scale Integr. (VLSI) Syst., vol. 11, no. 3, pp. 386–405, Mar.2003.
[19] W. Quddus, A. Jas, and N. A. Touba, "Configuration self-test in FPGAbasedreconfigurable systems," in Proc. ISCAS, 1999, pp. 97–100.
[20] C. Stroud, E. Lee, and M. Abramovici, "BIST based diagnostics ofFPGA logic blocks," in Proc. Int. Test Conf, 1997, pp. 539–547.
[21] C. Stroud, S. Konala, C. Ping, and M. Abramovici, "Built-in self-test oflogic blocks in FPGAs (Finally, a free lunch: Bist without overhead!),"in Proc. VLSI Test Symp., 1996, pp. 387–392.
[22] J. T. Chen, J. Khare, K.Walker, S. Shaikh, J. Rajski, and W. Maly, "Test response compression and bitmap encoding for embedded memories in manufacturing process monitoring," in Proc. Int. Test Conf. (ITC), 2001, pp. 258–267.
[23] Anita Aghaie, Mehran Mozaffari Kermani, Reza Azarderakhsh, "Reliable and Fault Diagnosis Architectures for Hardware and Software-Efficient Block Cipher KLEIN Benchmarked on FPGA", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 37, Issue 4, April 2018, pp 901 – 905.
[24] Gehad I. Alkady, Nahla A. El-Araby, M.B. Abdelhalim, H.H. Amer, A.H. Madian, "A fault-tolerant technique to detect and recover from open faults in FPGA interconnects", in the proceedings of 14th Biennial Baltic Electronic Conference (BEC), November 2015.