International Journal of Trend in Scientific Research and Development (IJTSRD)
International Open Access Journal
ISSN No: 2456 - 6470 | www.ijtsrd.com | Volume - 2 | Issue – 2

Fault Testing and Diagnosis of Sram based FBGA using Built-In-Self-Test-Architecture

Nagma. P
Student, Gnanamani College of Technology, Namakkal, Tamilnadu

Ramachandran. S
Assistant Professor, Gnanamani College of Technology, Namakkal, Tamilnadu

Sathishkumar. E
Assistant Professor, Gnanamani College of Technology, Namakkal, Tamilnadu

ABSTRACT

A new low-power (LP) scan-based built-in self test (BIST) technique is proposed based on weighted pseudorandom test pattern generation and reseeding. A new LP scan architecture is proposed, which supports both pseudorandom testing and deterministic BIST. During the pseudorandom testing phase, an LP weighted random test pattern generation scheme is proposed by disabling a part of scan chains. During the deterministic BIST phase, the design-for-testability architecture is modified slightly while short. Built in Self-Test (BIST) is a design technique that allows a circuit to test itself .The proposed method of a built-in self-test (BIST) design for fault detection and fault diagnosis of static-RAM (SRAM)-based field-programmable gate arrays (FPGAs). can test both the interconnect resources [wire channels and programmable switches (PSs)] and lookup tables (LUTs) in the configurable logic blocks (CLBs).The test pattern generator and output response analyzer are configured by CLBs in FPGAs. The target fault detection/diagnosis of the proposed BIST structure are open/short and delay faults in the wire channels, stuck on/off faults in PSs, andstuck-at-0/1 faults in LUTs. The testing process is performed by configuring the Test Pattern Generator (TPG), Output Response Analyzer (ORA) and Block under Test (BUT) in each test block.

Keywords: *low-power, fault detection and fault diagnosis, field-programmable gate arrays*

INTRODUCTION

1.1 FIELD-PROGRAMMABLE GATE ARRAYS

Field programmable gate arrays (FPGA) have been widely utilized in digital system design; FPGAs offer programmability at relatively low development cost and good performance. An FPGA consists of logic and interconnect resources that permit to configure an uncommitted chip into the desired functions for different applications. The common FPGA architecture consists of a two-dimensional array of configurable logic blocks (CLBs), a programmable interconnect (made of nets and switches) and programmable input/outputs (IOs). Currently, the programmable interconnect accounts for more than 80 percent of the FPGA programmable resources (usually stacked up in eight metal layers). Therefore, nets in the interconnect resources are very vulnerable to faults both at manufacturing as well as run time. For interconnect resources, FPGA testing methodologies are broadly classified as application-dependent and application-independent. Application-independent testing (also known as manufacturing test) utilizes nearly exhaustive test sets such that all programmable resources are tested for faults.

Field-Programmable Gate Arrays (FPGAs) to implement complex logic functions in digital applications has become increasingly common. FPGAs are regular structures of logic modules that communicate through an interconnected architecture of lines and switches. The logic modules and the interconnect structures are programmed to select a particular function of each logic module and specific interconnect paths to realize the global function of the FPGA.

FPGAs are generally classified into two major types. One is one-time programmable FPGAs such as an antiques type. The other is unlimited reprogrammable FPGAs such as a static RAM (SRAM)-based FPGA. In SRAM-based FPGAs, logic elements and programmable switches (PSs) can be reprogrammed by loading a configuration bit stream, giving FPGAs incredible flexibility that is enough to implement any digital circuit on the same piece of silicon. Since FPGA results in a shorter time to market and increased flexibility for systems using logic circuits, many applications have been developed to make the best use of FPGA reprogram ability.

Currently, very-large-scale integration (VLSI) technology keeps increasing circuit integration by ever greater degrees, and the rapid development in packaging technology greatly reduces the controllability and observability of internal nodes. This significantly complicates the testing of the system. Generally, FPGAs can be configured in an incredibly large number of ways. From the manufacturing testing point of view, it must be ensured that the part is functional under all configurations. Since the reconfiguration time (the time to load a configuration into an FPGA) ranges from milliseconds to entire seconds, depending on the size of the FPGA, it is impossible to map all possible configurations during a test. Therefore, a set of test configurations (TCs) must be developed to ensure that the part is defect free, regardless of any particular user application. In other words, FPGA testing is important to ensure the reliable performance of FPGA devices. In particular, SRAM based FPGA testing has been attracting the interest of a large number of researchers in the last decade since the complexity and the size of FPGA devices have increased.

Several previous works extensively studied the testing of FPGAs assuming that only configurable logic blocks (CLBs) can be faulty. Similarly, others studied k2, and k3 the testing of only the interconnect resources, assuming that CLBs were fault free. To date, however, there has been relatively little research conducted on both interconnect resource and CLB testing. Although the synthesis algorithm for self-checking the combinational logic of FPGAs can detect the faults within the combinational functional block of a CLB and on the interconnect lines connecting the functional blocks, the approach is short of the ability of fault diagnosis.

In contrast, conventional BIST approaches introduce both area overhead (typically between 10% and 30%) and delay penalties. The BIST approach can be applied to any in-circuit reprogrammable SRAM-based FPGAs. The only cost will be the additional memory for storing the data required to reconfigure the FPGA.

1.2 SRAM-BASED FPGA ARCHITECTURE

SRAM-based FPGAs are of special interest due to their compatibility with mainstream integrated circuit fabrication technology, as well as because of their wide use in practical applications. The SRAM-based FPGA consists of an array of n × n CLBs and local/global interconnect resources. The CLBs can be programmed with configuration cell data to generate logical functions. The set of all configuration cell data makes up an FPGA configuration. The basic internal architecture of a CLB is made up of three components: 1) lookup tables (LUTs); 2) multiplexers (MUXs); and 3) D-type flip-flops (DFFs). In figure.1, the LUTs implement either any logical function with k1 inputs or RAM. Every CLB is configured with configuration cells C1, . . . ,Cn. The number of outputs of LUTs is assumed to be k3, and the inputs k2 directly drive some MUXs or DFFs, producing the CLB outputs. Note that parameters k1,depend on the FPGA type.

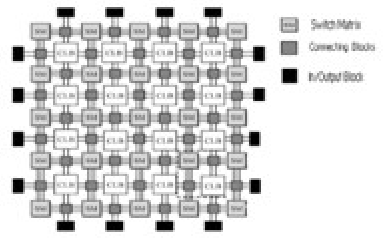

Fig. No 1: Fpga Architecture

The local interconnects are associated with CLBs, including wire segments and connecting blocks. Note that a connecting block contains some programmable-interconnect-point PSs (PIP-PSs) and multiplexer PSs (MUX-PSs) to bring signals into and out of CLBs. On the other hand, wire segments and programmable cross-point PSs (PCP-PSs) within the switch matrix (SM) in global interconnects form horizontal and vertical routing channels that connect signals between CLBs.

1.3 TESTING CONSIDERATIONS

The FPGA test procedure is a two-part process. Each TC must be first loaded into the FPGA device, and then, test vectors must be applied. This process is repeated for a number of TCs. The PIP-PSs and MUX-PSs in the connecting block and the PCP-PSs in the SM are the essential components for programming TCs under FPGA testing. The SRAM cell can be programmed to open or close the pass transistor. An MUX-PS, illustrated as a set of unidirectional triangles over the horizontal and vertical wire segments, functions as a many-to-one MUX. An MUX-PS allows one of the inputs to be routed to the output for certain selection signals.

The selection logic can only be set through the configuration bits during the configuration stage. In other words, a PCP-PS connects the wire segments in the west– east (W–E), north–south (N–S), north–west (N–W), south–west (S–W), north–east (N–E), and south–east (S–E) directions.

Generally, the TCs are fault graded. If the desired fault coverage is not achieved, test engineers must develop additional test vectors and configurations to increase the fault coverage. In other words, the testing of an FPGA chip poses a challenge to test engineers. Different configurations (programming's) of the FPGA are required, and so, certain questions arise. It is extremely important to bear in mind the high cost of changing the configuration due to reprogramming costs. The configuration loading process is performed through a serial access, and so, this process is really time consuming. Consequently, when test generation for an FPGA is performed, the main objective is to minimize the number of TCs.

Field-programmable gate arrays (FPGAs) are used to implement complex logic functions in digital applications has become increasingly common. Due to this so many faults are occurred in interconnect resources and CLBs to overcome those faults a BIST is designed in this project.

LITERATURE SURVEY

2.1 A NOVEL TECHNIQUE FOR PEAK- AND AVERAGE-POWER REDUCTION IN SCAN-BASED BIST

Abdallatif S. Abu-Issa and Steven F. Quigley

A novel low-transition linear feedback shift registers (LFSR) that is based on some new observations about the output sequence of a conventional LFSR. It's composed of an LFSR and a 2 × 1 multiplexer. When used to generate test patterns for scan-based built-in self-tests, it reduces the number of transitions that occur at the scan-chain input during scan shift operation by 50% when compared to those patterns produced by a conventional LFSR. Hence, it reduces the overall switching activity in the circuit under test during test applications. The BS-LFSR is combined with a scan-chain-ordering algorithm that orders the cells in a way that reduces the average and peak power (scan and capture) in the test cycle or while scanning out a response to a signature analyzer. These techniques have a substantial effect on average- and peak-power reductions with negligible effect on fault coverage or test application time. Experimental results on ISCAS'89 benchmark circuits show up to 65%and 55% reductions in average and peak power, respectively.

The design for low power has become one of the greatest challenges in high performance very large scale integration(VLSI) design. As a consequence, many techniques have been introduced to minimize the power consumption of new VLSI systems. However, most of these methods focus on the power consumption during normal mode operation, while test mode operation has not normally been a predominant concern. However, it has been found that the power consumed during test mode operation is often much higher than during normal mode operation. This is because most of the consumed power results from the switching activity in the nodes of the circuit under test (CUT), which is much higher during test mode than during normal mode operation. Several techniques that have been developed to reduce the peak and average power dissipated during scan-based tests can be found in paper. A direct technique to reduce power consumption is by running the test at a slower frequency than that in normal mode. This technique of reducing power consumption, while easy to implement, significantly increases the test application time. Furthermore, it fails in reduce in speak-power consumption since it is independent of clock frequency Another category of techniques used to reduce the power consumption in scan-based built-in self-tests (BISTs) is by using scan chain ordering techniques.

These techniques aim to reduce the average-power consumption when scanning in test vectors and scanning out captured responses. Although these algorithms aim to reduce average-power consumption, they can reduce the peak power that may occur in the CUT during the scanning cycles, but not the capture power that may result during the test cycle (i.e., between launch and capture)

2.2 WEIGHTED PSEUDORANDOM HYBRID BIST

Abhijit Jas, C. V. Krishna, and Nur A. Touba,

Test data-compression scheme that is a hybrid approach between external testing and built-in self-test (BIST). Three levels of compression are used to greatly reduce test costs. Experimental results show that the proposed scheme reduces tester storage requirements and tester bandwidth requirements by orders of magnitude compared to conventional external testing, but requires much less area overhead than a full BIST implementation providing the same fault coverage. No test points or any modifications are made to the function logic. The paper describes the proposed hybrid BIST architecture as well as two different ways of storing the weight sets, which are an integral part of this scheme.

A new test data compression scheme is presented, which is a hybrid approach between BIST and external tested. The term "hybrid BIST" will be used in this paper to classify any scheme that involves combining external data from the tester along withiest hardware on the chip to provide a hybrid test solution for a particular module or core. A hybrid BIST approach reduces the test data stored on the tester compared with full external testing, but does not require as much hardware overhead as full BIST. A simple approach for hybrid BIST is to use STUMPS architecture to apply pseudorandom patterns to detect the random pattern testable faults, and then use deterministic scan vectors from the tester to detect the hard faults.

There have been a couple of case studies on using this approach for large industrial designs. The case study in was done on the Motorola PowerPC microprocessor core, and the study in was done on large ASIC designs. In the reduction in external test storage requirements after using 500 K BIST patterns was around 30%. In test points were inserted, but the reduction in test storage requirements after 262 K BIST patterns still ranged only from 35% to 55%. What these results indicate is that most of the vectors in a deterministic test set target hard faults which are missed by pseudorandom patterns. So a straightforward hybrid BIST approach where pseudorandom vectors are applied with BIST hardware followed by deterministic vectors from the external tester, can only achieve a limited reduction infester storage requirements, generally not an order of magnitude reduction. There are several other approaches that can be classified as hybrid BIST approaches. In a hybrid BIST approach was proposed where some of the scan chains in STUMPS architecture are filled with deterministic test data from the tester while the rest of the scan chains are filled from the pseudorandom pattern generator (PRPG).

The set of scan chains receiving deterministic data is rotated in a round-robin fashion. This approach was applied to the Motorola PowerPC microprocessor core. Results indicated that the test storage requirements could be reduced by around 50% with this approach compared with 31% as was reported in for using fully pseudorandom patterns followed by fully deterministic patterns.

2.3 AN APPLICATION SPECIFIC RECONFIGURABLE ARCHITECTURE FOR FAULT TESTING AND DIAGNOSIS: A SURVEY

A.R Kasetwar, Gaurav Kumar, S. M. Gulhane

Now a day's many VLSI designers are implementing different applications on real time with the use of FPGAs. Although they are working efficiently, they are not achieving their expected goals. This is only because of the faults which are occurring in the FPGA at the runtime of the application. Those faults are remaining in the circuitry as there is no provision for removal of those faults at application level. So there is a great need of detection & removal of faults. Mainly Interconnect faults, Logical Faults and Delay are the faults which reduces the performance of FPGA. Although the manufacturers are trying to decrease the fault present in the FPGA, it is very necessary to remove those faults at run time of the particular application. This paper includes the occurrence of different faults and various methods to remove those faults.

Fault diagnosis has particular importance in the context of field programmable gate arrays (FPGAs) because faults can be avoided by reconfiguration at almost no real cost. The main faults in the FPGAs are interconnects, logic blocks and faults related to delay of an arbitrary design. As FPGAs works properly, all three faults should be removed. FPGA testing is of two types an application dependent and application independent. The testing process for both is different. The time required to test an application dependent testing is less as it focuses only on some specific part whereas the application independent testing tests complete FPGAs .Faults related to logical block, interconnected faults and delay faults are the problems for the FPGAs user.

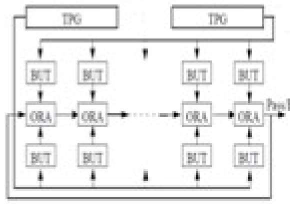

Fig. No 4 : BIST Architecture TPG, But And ORA Connection.

They have made a detail survey of number of methods for fault diagnosis from this survey they came to know that for interconnect diagnosis, the method is better as compared to other method because all the single faults and multiple are uniquely identified and removed. For logic blocks diagnosis, BISD approach is selected because in it multiple faults are uniquely identified in a single test configuration with a fixed test time. All other methods are limited only for faulty CLBs and single faults. It is used for both manufacturing as well as user configuration test with the guarantee that the maximum delays along the tested path will not exceeds the clock period during the normal operation

2.4 A NOVEL HEURISTIC METHOD FOR APPLICATION-DEPENDENT TESTING OF A SRAM-BASED FPGA INTERCONNECT

T. Nandha Kumar, Senior Member, IEEE, and Fabrizio Lombardi, Fellow, IEEE

A new method for generating configurations for application-dependent testing of a SRAM-based FPG Interconnects. This method connects an activating input to multiple nets, thus generating activating test vectors for detecting stuck-at, open, and bridging faults. This arrangement permits a reduction in the number of redundant configurations, thus also achieving are diction in test time for application-dependent testing at full fault coverage. As the underlying solution requires an exponential complexity, a heuristic algorithm that is polynomial and greedy in nature (based on sorting) is used for net selection in the configuration generation process. It is proved that this algorithm has an execution complex (where L is the number of LUTs in the design). The proposed method requires at most log2ðM þ 2Þ configurations (where M denotes the number of activating inputs) as Walsh coding is employed.

For interconnect resources, current FPGA testing methodologies are broadly classified as application dependent and application independent. Application-independent testing (also known as manufacturing test) utilizes nearly exhaustive test sets such that all programmable resource sari tested for faults. An application-dependent method tests those resources that are utilized by a specific design implemented on an FPGA. This type of testing method has drawn particular interest as it can implement adaptive fault tolerance and self-diagnosis as well as self-repair can be utilized. So, it is important to shorten the test time as periodical tests are used to identify and locate a faulty resource by complete (100 percent) coverage; execution complexity of the test generation process is not as important(provided it remains polynomial) because the configurations are established once for a given application. A possible approach to shorten test time with complete coverage is to reduce the number of test configurations.

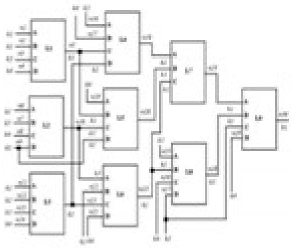

Fig.No 5 :Activating Input Assignment Using Application-Dependent Testing

In the algorithmic method for application-dependent testing of a SRAM-based FPGA interconnects. The proposed method relies on generating and utilizing so-called activating inputs connected through multiple nets with Walsh coding. This algorithmic-based method detects all stuck-at, open, and pair wise bridging faults in the interconnect resources of an FPGA. Analysis and simulation has shown that the heuristic criterion used in the proposed method results in an efficient generation of test configurations. At most configurations are required (where M denotes the number of activating inputs) in theory; however, an assessment of 25 ISCAS89sequential benchmarks has shown that the required number of configurations is very small (three or four) and therefore almost independent of the design size and the number of inputs of a LUTs in the FPGA. Test configurations have been generated using an algorithmic method whose execution is in (where L is the number of LUTs in the design); this method utilizes net sorting to assign activating inputs and finding the test vectors.

2.5 HIGH RESOLUTION APPLICATION SPECIFIC FAULT DIAGNOSIS OF FPGAS

Mehdi B. Tahoori, Senior Member, IEEE

High resolution diagnosis plays a critical role in silicon debugs and yield improvement. Application-dependent diagnosis is also a key component in online testing and adaptive computing. In this paper, a new technique for high resolution localization of faults in the interconnects and logic blocks of an arbitrary design implemented on a field-programmable gate array (FPGA)is presented. This work is complementary to application-independent detection methods for FPGAs. This technique can uniquely identify any single bridging, open, or stuck-at fault in the interconnects well as any single functional fault, a fault resulting change in the truth table of a function, in the logic blocks. Then umber of test configurations for interconnect diagnosis is logarithmic to the size of the mapped design, whereas logic diagnosis is performed in only one test configuration with less than 5% overhead of built-in self diagnosis. These techniques have been further extended for multiple fault diagnosis.

Application-dependent diagnosis is also a key player in silicon debug process. Once a particular configuration (e.g., a test configuration at the manufacturing test or an application configuration in the field) fails, the location of defective resources needs to be precisely identified for failure mode analysis (FMA)and yield improvement. An effective application-dependent fault localization (diagnosis) method can reduce the overall silicon debug time and improve its precision and quality. During system operation, application-dependent test and diagnosis are very crucial in online self-repair schemes for fault tolerant applications. 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 resources used for that particular design, rather than all FPGA resources.

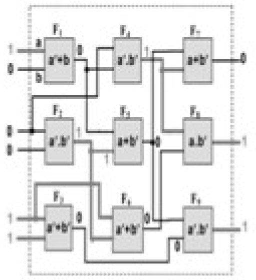

Fig .No 6 : Logic Network Of Single-Term Functions

Application-dependent diagnosis techniques for faults in interconnect and logic blocks of an arbitrary design mapped into an FPGA are presented. For interconnect diagnosis, multiple faults (open, stuck-at, or bridging fault) can be uniquely identified. As shown in the paper, the number of total test configurations for diagnosis of interconnect is logarithm micro the size of the design. For logic block diagnosis ,a BISD approach is presented in which multiple faults can be uniquely identified in only one test configuration. This method can be used for defect tolerance by the manufacturer in order to increase the manufacturing yield, i.e., as a part of application-specific FPGA (ASFPGA) test flow, or in the online self-repair schemes for fault tolerant applications.

EXISTING SYSTEM

3.1 DFT ARCHITECTURE

As shown in Fig. 1, the scan-forest architecture is used For pseudorandom testing in the first phase. Each stage of the phase shifter (PS) drives multiple scan chains, where all scan chains in the same scan tree are driven by the same stage of the PS. Unlike the multiple scan-chain architecture used in the previous methods, the scan-forest architectures adopted to compress test data and reduce the deterministic test data volume. Separate weighted signals e0, e1. . . and en are assigned to all scan chains in the weighted pseudorandom testing phase (phase = 0), as shown in

Fig., which is replaced by the regular test in the deterministic BIST phase (phase = 1). Each scan-in signal drives multiple scan chains, as shown in Fig. 1, where different scan chains are assigned different weights. This technique can also significantly reduce the size of the PS compared with the multiples can-chain architecture where each stage of the PS drives one scan chain. The compactor connected to the combinational part of the circuit is to reduce the size of the MISR. The size of the LFSR needed for deterministic BIST depends on the maximum number of care bits of all deterministic test vectors for most of the previous deterministic BIST methods. In some cases, the size of the LFSR can every large because of a few vectors with a large number of care bits even when a well-designed PS is adopted. This may significantly increase the test data volume in order to keep the seeds. This problem can be solved by adding a small number of extra variables to the LFSR or ring generator without keeping a big seed for each vector.

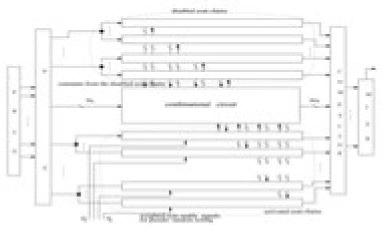

Fig. No 7 : Weighted Pseudorandom Test Generator For Scan-Tree-Based Lp BIST.

3.2. WEIGHTED PSEUDORANDOM TEST PATTERN GENERATION

Our method generates the degraded sub circuits for all subsets of scan chains in the following way. All PPIs related to the disabled scan chains are randomly assigned specified values (1 and 0). Note that all scan flip flops at the same level of the same scan tree share the same PPI. For any gate, the gate is removed if its output is specified; the input can be removed from a NAND, NOR, AND, and OR gates if the input is assigned a no controlling value and it has at least three inputs. For a two-input AND or OR gate, the gate is removed if one of its inputs is assigned a no controlling value. For a NOR or NAND gate, the gate degrades to an inverter if one of its inputs is assigned a no controlling value.

For an XOR or NXOR gate with more than three inputs, the input is simply removed from the circuit if one of its inputs is assigned value 0; the input is removed if it is assigned value 1, an XOR gate changes to an NXOR gate, and an NXOR gate changes to an XOR gate. For an XOR gate with two inputs, and one of its inputs is assigned value 0, the gate is deleted from the circuit. For a two-input NXOR gate, the gate degrades to an inverter. If one of its inputs is assigned value 1, a two-input XOR gate degrades to an inverter. If one of its inputs is assigned value 1, a two-input NXOR gate can be removed from the circuit.

3.3 LOW-POWER DETERMINISTIC BIST

We use the same LFSR for both pseudorandom pattern generation and deterministic phases. First, we propose a new algorithm to select a proper primitive polynomial; after that the LP deterministic BIST and LP reseeding schemes are presented.

3.3.1 SELECTING A PRIMITIVE POLYNOMIAL AND THE EXTRA VARIABLE NUMBER

Some extra variables are injected just like EDT.A new scheme to select the size of the LFSR. In this number of extra variables simultaneously minimize the amount of deterministic test data. Usually, a small LFSR constructed by a primitive polynomial is sufficient when a well-designed PS is adopted in the pseudorandom testing phase. In our method, a combination of a small LFSR and the PS from is used to generate test patterns in the pseudorandom testing phase. The weighted test-enable signal-based pseudorandom test generator generates weighted pseudorandom test patterns. The size of the LFSR is not determined by the maximum number of care bits for any deterministic test vector. That is, the same LFSR is used for both phases. For any degree less than 128, it is computationally feasible to generate enough primitive polynomials in reasonable time, out of which one (whose degree is equal to the maximum number of care bits in the deterministic vectors) can be selected to encode all deterministic test vectors. The tool that we used to generate primitive polynomials can only handle polynomials up to degree 128 of the word-length limit of the computer.
However, only very small LFSRs are used for all circuits according to all experimental results (no more than 30)

3.3.2 LOW-POWER DETERMINISTIC BIST AND RESEEDING

An effective seed encoding scheme is used here to reduce the storage requirements for the deterministic test patterns of the random-pattern-resistant faults. The encoded seed is shifted into the LFSR first. A deterministic test vector is shifted into the scan trees that are activated by the gating logic, where each scan-in signal drives a number of scan trees, and only one of the scan trees driven by the same scan-in signal is activated. The extra variables are injected into the LFSR when the seed is shifted into the activated scan trees. The gating logic, as shown in Fig. 1, partitions scan trees into multiple groups. The first group of scan trees is disabled after they have received the test data. The second group of scan trees is activated simultaneously, and all other scan trees are disabled. The seed can be stored in an extra shadow register, which is reloaded to the LFSR in a single clock cycle. The scan shift operations are repeated when the extra variables are injected into the LFSR. This process continues until all scan trees have received test data. The outputs of all scan chains, which are driven by the same clock signal, are connected to the same response compactor during the deterministic BIST phase. This offers additional flexibility for test encoding. The test responses of the previous test vector can be shifted out with only a few clock cycles (corresponding to the depth of the scan trees in the pseudorandom testing phase). For scan chain architecture, the number of clock cycles needed to shift-out test responses of the previous deterministic test vector is much larger. The proposed LP tree-based architecture makes the reseeding scheme much easier to implement.

LIMITATIONS:

- BSF detects all possible stuck-at and bridging faults by utilizing the all zeros' vector and a walking-1 test set.
- Open faults cannot be guaranteed to be located.

PROPOSED SYSTEM

4.1 FPGA-BIST DESIGN

BIST techniques in general are associated with high performance; they are also associated with high area overhead incurred by on-chip test hardware. However, the BIST overhead is not an issue for FPGA BIST because the test hardware is easily reconfigured by inserting and removing test pattern generators (TPGs) and ORAs. This is particularly important for the testing of FPGAs. The testing strategy of the proposed FPGA BIST structure is to configure groups of ten CLBs into a test block, as illustrated in figure.

In each test block, four CLBs are configured as a TPG to generate the addresses for test patterns. Additionally, two CLBs are configured as an ORA for comparison with each output of the block under test (BUT) to observe the test results. The global/local interconnect resources and CLBs in a BUT, which are configured by four CLBs in a test block, are then sequentially tested. To guarantee the testing of all global/local interconnects resources and CLBs, the FPGA has to be reconfigured to shift the test blocks for testing. The test processes of the proposed FPGA BIST structure are simultaneously performed by a BIST controller, which repeatedly reconfigures the test blocks for testing.

Briefly, the testing processes can be summarized in the following steps.

1. Reconfigure the FPGA to create test blocks.
2. Program the TCs.
3. Initiate the TS for global/local interconnect resources and CLBs.
4. Generate the test vectors.
5. Analyze the test results.

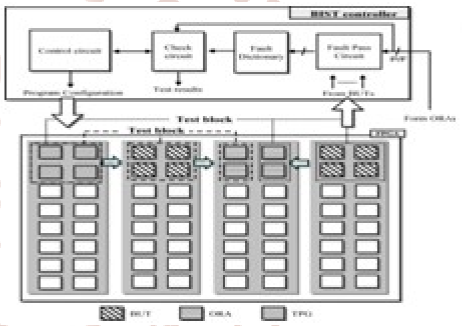

Fig .No 8 : FPGA-BIST Structure

In other words, the test blocks are first (re)configured by the BIST controller. Second, the TCs should be reconfigured for global/local interconnect resource and CLB testing. Then, the LUT-based method is used to configure the TPG and ORA to generate the test vectors. Finally, the test results are analyzed.

4.2 TC CONFIGURATIONS

The global/local interconnect resource and CLB tests rely on the in-system reprogram ability of FPGAs. Thus, separate testing treatments are needed since the global interconnects, local interconnects, and CLBs of an FPGA are fundamentally different in nature. The testing of global interconnects is achieved by configuring PCP-PSs in SMs and horizontal and vertical wire segments to form long buses, and then, their integrity is verified. The formation of the buses does not use any CLB logic. On the contrary, when local interconnects are under test, the CLBs must be part of the test circuitry, since local interconnects can only be tested indirectly by applying tests to the inputs of a CLB and observing test responses at the CLB outputs. To ensure the testing of the entire FPGA and reduce the testing time, the TCs have to be carefully scheduled first, particularly the SMs in the global interconnects.

4.3 FAULT MODELS

Design a BIST structure for both CLB and interconnect resource testing in SRAM-based FPGAs. The faults in this paper can be categorized into four major groups, namely, open/short, stuck-on/off, stuck-at-0/1, and interconnect delay faults. The stuck-on/off faults appear in the pass transistor of PIP-PSs or MUX-PSs in local interconnects, while open/short faults occur on PCP-PSs or wire segments in global interconnects. Significantly, the delay fault is presented with a path under test (PUT). On the other hand, the stuck-at-0/1 faults can be found in the LUTs of CLBs.

Figure shows cases of PIP-PSs and MUX-PSs with stuck on/off faults in the local interconnects. Note that a stuck-on/off fault causes the pass transistor in PIP-PSs or MUX-PSs to be permanently on/off, regardless of the value of the SRAM cell controlling the pass transistor in PIP-PSs or MUX-PSs. An open fault in the global interconnect is a disconnection of any wires, while a short fault indicates a bridging between two wires.

Figure (b) illustrates cases of wire open/short faults. Furthermore, Figure (b) shows cases of PCP-PS open/short faults that occur when there is a stuck-off fault and a stuck-on fault in the PIP-PS of the connectable and no connectable directions of wire segments, respectively. Moreover, since the TCs for interconnect resource and CLB testing are the same and the TPG and ORA of the proposed BIST design are built by using the LUTs, the faults in a CLB only consider the stuck-at-0/1faults on every AM cell in LUTs [see figure (c)]. In other words, for an LUT, the fault can occur in any one of the memory cells, making it incapable of storing the correct logic value (an LUT has a single-bit output, and therefore, this value is either 0 or 1). Thus, the stuck-at-1 or the stuck-at-0 fault may occur at a memory cell. Note that there is no need to separately consider the stuck-at fault model in interconnects, since these faults can be modeled as short circuits to power supply and ground lines. On the other hand, by setting the clock period to the specification time and generating a transition to go through the PUT, we can be determine whether the PUT is fault free or not.

Figure (d) shows an example of testing a PUT. If the two DFFs initially store logic 0 and generate rising-transition propagation from A to B, then logic 1 in DFF2 can be obtained in the fault-free situation. Otherwise, the logic value of DFF2 will remain 0 after a specified time if the PUT is faulty.

TYPES OF FAULT MODELS:

Fault models are,

1) Wire open fault:
A disconnection occurs on any wires in the global interconnect.
2) Wires short fault:
A bridge occurs between two wires in the global interconnect.
3) LUT stuck-at-0 fault:
The RAM cell value of an LUT in the CLB is always 0.
4) LUT stuck-at-1 fault:
The RAM cell value of an LUT in the CLB is always 1.
5) Interconnect delay fault:
A transition is propagated from one end of a circuit, and the result is captured from the other end after a specified time (test clock period)

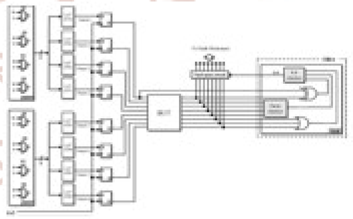

Fig .No 10: Scheme of A BIST Structure in a Test Block.

ADVANCED FPGA-BIST DESIGN

❖ STo test an SRAM-based FPGA using a BIST technique, it has to be configured into TCs, and then, test vectors, input values, and primary inputs must be applied in each configuration.
❖ The BIST method involves configuring one part of the FPGA to undergo testing and configuring the other parts to generate test vectors and to analyze test results.

- ❖ The sub circuits being tested and the resources of the FPGA change roles so that the entire FPGA is eventually tested. This way, testability is achieved without any overhead, since the BIST logic will "disappear" when the circuit is reconfigured for its normal system operation.
- ❖ Although BIST techniques in general are associated with high performance, they are also associated with high area overhead incurred by on-chip test hardware. However, the BIST overhead is not an issue for FPGA BIST because the test hardware is easily reconfigured by inserting and removing test pattern generators (TPGs) and ORAs.

APPLICATIONS:

- ➢ Weapons
- ➢ Avionics
- ➢ Safety-Critical devices
- ➢ Automotive use
- ➢ Computers
- ➢ Unattended machinery
- ➢ It may be implemented in Space Research Applications.
- ➢ For most advanced Self Intelligent Robotic technology.
- ➢ Fully automatic digital controlled equipment.
- ➢ It may be also implemented in Advanced automobiles

SYSTEM SPECIFICATIONS

The simulation and Hardware implementation can be done by using Models, Xilinx ISE.

5.1 INTRODUCTION TO HDL

VHDL is a language for describing digital electronic systems. It arose out of the United States government's very high speed integrated circuits (VHSIC) program, in the course of this program. It became clear that there was a need for a standard language for describing the structure and function of integrated circuits(IC's). Hence the VHSIC hardware description language (VHDL) was developed. It was subsequently developed further under the auspices of the Institute of Electrical and Electronics Engineers (IEEE) and adopted in the form of the IEEE standard 1076. VHDL is designed to fill a number of needs in the design process.

- ➢ First, it allows description of the structure of a system which includes decomposition into subsystems and interconnects of those subsystems.
- ➢ Second it allows the specification of the function of a system to be stimulated before being manufactured, so that designers can quickly compare alternatives test for correctness without the delay and expense if hardware prototyping.
- ➢ Third it allows the detailed structure of a design to be synthesized from a more abstract specification, allowing designers to concentrate on more strategic design and reducing time to market.
- ➢ Modeling for simulation and synthesis is a vital part of a range of levels of abstraction, from gate levels up to algorithmic and architectural levels. It will continue to play an important role in the design future silicon-based systems.

Very high speed integrated circuit hardware description language (VHDL) can be used to model digital systems and introduce some of the basic concepts underlying the language.

HDL LANGUAGE

HDL language involved with this topics

- ➢ Digital Design with HDL.
- ➢ Emergence of HDLs.
- ➢ Popularity of Virology HDL.

DIGITAL DESIGN WITH HDL

Digital circuit design has evolved rapidly over the last 25 years. The earliest digital circuits were designed with vacuum tubes and transistors. Integrated circuits were then invented where logic gates were placed on a single chip. The first integrated circuit (IC) chips were SSI (Small Scale Integration) chips where the gate count was very small. As technologies became sophisticated, designers were able to place circuits with hundreds of gates on a chip. These chips were called MSI (Medium Scale Integration) chips. With the advent of LSI (Large Scale Integration), designers could put thousands of gates on a single chip. At this point, design processes started getting very complicated, and designers felt the need to automate these processes. Computer Aided Design (CAD) techniques began to evolve. Chip designers began to use circuit and logic simulation techniques to verify the functionality of building blocks of the order of about 100 transistors. The circuits were still tested on the breadboard, and the layout was done on paper or by hand on a graphic computer terminal.

With the advent of VLSI (Very Large Scale Integration) technology, designers could design single chips with more than 100,000 transistors. Because of the complexity of these circuits, it was not possible to verify these circuits on a breadboard. Computer-aided techniques became critical for verification and design of VLSI digital circuits. Computer programs to do automatic placement and routing of circuit layouts also became popular. The designers were now building gate-level digital circuits manually on graphic terminals. They would build small building blocks and then derive higher-level blocks from them. This process would continue until they had built the top-level block. Logic simulators came into existence to verify the functionality of these circuits before they were fabricated on chip.

5.2 EMERGENCE OF HDLS

For a long time, programming languages such as FORTRAN, Pascal, and C were being used to describe computer programs that were sequential in nature. Similarly, in the digital design field, designers felt the need for a standard language to describe digital circuits. Thus, Hardware Description Languages (HDLs) came into existence. HDLs allowed the designers to model the concurrency of processes found in hardware elements. Hardware description languages such as Virology HDL and VHDL became popular. Virology HDL originated in 1983 at Gateway Design. Even though HDLs were popular for logic verification, designers had to manually translate the HDL-based design into a schematic circuit with interconnections between gates. The advent of logic synthesis in the late 1980s changed the design methodology radically. Digital circuits could be described at a register transfer level (RTL) by use of an HDL. Thus, the designer had to specify how the data flows between registers and how the design processes the data. The details of gates and their interconnections to implement the circuit were automatically extracted by logic synthesis tools from the RTL description.

Thus, logic synthesis pushed the HDLs into the forefront of digital design. Designers no longer had to manually place gates to build digital circuits. They could describe complex circuits at an abstract level in terms of functionality and data flow by designing those circuits in HDLs. Logic synthesis tools would implement the specified functionality in terms of gates and gate interconnections. HDLs also began to be used for system-level design. HDLs were used for simulation of system boards, interconnect buses, FPGAs (Field Programmable Gate Arrays), and PALs (Programmable Array Logic). A common approach is to design each IC chip, using an HDL, and then verify system functionality via simulation.

5.3 POPULARITY OF VERILOG HDL

Virology HDL has evolved as a standard hardware description language. Virology HDL offers many useful features for hardware design. Virology HDL is a general-purpose hardware description language that is easy to learn and easy to use. It is similar in syntax to the C programming language. Designers with C programming experience will find it easy to learn Virology HDL. Virology HDL allows different levels of abstraction to be mixed in the same model. Thus, a designer can define a hardware model in terms of switches, gates, RTL, or behavioral code. Also, a designer needs to learn only one language for stimulus and hierarchical design. Most popular logic synthesis tools support Virology HDL. This makes it the language of choice for designers. All fabrication vendors provide Virology HDL libraries for post logic synthesis simulation. Thus, designing a chip in Virology HDL allows the widest choice of vendors. The Programming Language Interface (PLI) is a powerful feature that allows the user to write custom C code to interact with the internal data structures of Virology. Designers can customize a Virology HDL simulator to their needs with the PLI.

HIERARCHICAL MODELING CONCEPTS OF VERILOG

CONCEPT OF A 'MODULE'

- A module is a basic building block in Virology.
- It can be an element or a collection of lower-level design (macro or leaf cells) blocks.
- Provides functionality through its port interface.

DECLARATION OF A MODULE

- A module in Virology is declared using the keyword module.
- A corresponding keyword end module must appear at the end of the module.
- Each module must have a module name, which act as the identifier.

- A module can have an optional port list which describes the input and output terminals of the module.

DIFFERENT ABSTRACTION LEVELS

- Internals of each module can be defined at four abstraction levels.

- Behavioral or Algorithmic level
- Dataflow level.
- Gate or structural level.
- Switch level.

BEHAVIORAL OR ALGORITHMIC

At this level, a module is implemented in terms of desired algorithm or behavior without concern for the hardware implementation details.

```
// Behavioral or Algorithmic level
Module and gate (a, b, out);
// I/O port declaration
input a, b;
output out;
// Variable declaration
rag out;
// always behavioral statement
always @ (a or b)
out = a & b;
endmodule
```

SDATAFLOW LEVEL

At this level, a module is designed by specifying the data flow between hardware registers and how the data is processed in the design.

```
// Data flow level
Module and_ gate (a, b, out);
// I/OPort declaration
input a, b;
output out;
// Dataflow assign statement
assign out = a & b;
end module
```

Gate or Structural level
At this level, a module is implemented in terms of logic gates and interconnections between these gates.

```
// Gate level or structural level
Module and _gate (a, b, out);
// I/O port declaration
input a, b;
output out;
// Verilog primitive gate instantiation
and a1(out ,a, b);
end module
```

SWITCH LEVEL

This is the lowest level of abstraction in which, a module can be implemented in terms of switches, storage nodes and the interconnections between them.

```
// Switch level
Module nor _gate (out, a, b);
// I/O port declaration
input a, b;
output out;
// internal wires
wire c;
// set up power and ground lines
supply1pwr;
supply0gnd;
// instantiate pmos switches
pmos (c, pwr, b);
pmos (out, c, a);
end module
```

MODELSIM

This section describes the basic procedure for simulating Models.

Library
A library is a location on your file system where Models stores data to be used for Simulation. Models use one or more libraries to manage the creation of data before it is needed for use in simulation. A library also helps to streamline simulation invocation. Instead of compiling all design data each time you simulate, Models uses binary pre-compiled data from its libraries. For example, if you make changes to a single Virology module, Models recompiles only that module, rather than all modules in the design.

5.4 MAPPING THE LOGICAL WORK TO THE PHYSICAL WORK DIRECTORY

Step 1: VHDL uses logical library names that can be mapped to Models library directories. If libraries are not mapped properly, and you invoke your simulation, necessary components will not be loaded and simulation will fail. Similarly, compilation can also depend on proper library mapping. By default, Models can find libraries in your current directory (assuming they have the right name), but for it to find libraries located elsewhere, you need to map a logical library name to the pathname of the library. You can use the GUI (Library Mappings with the GUI, a command (Library Mapping from the Command Line), or a project (Getting Started with Projects to assign a logical name to a design library.

Step 2: Compile the Design
To compile a design, run one of the following Models commands, depending on the language used to create the design:
Compiling Virology (VLOG):
The log command compiles Virology modules in your design. You can compile Virology files in any order, since they are not order dependent. See Virology Compilation for details.
Compiling VHDL (VCOM):
The vcom command compiles VHDL design units. You must compile VHDL files in the order necessitate to any design requirements. Projects may assist you in determining the compile order.

Step 3: Load the Design for Simulation
Running the Vim Command on the Top Level of the Design
After you have compiled your design, it is ready for simulation. You can then run the vim command using the names of any top-level modules (many designs contain only one top-level module).
Using Standard Delay Format Files
You can incorporate actual delay values to the simulation by applying standard delay format (SDF) back-annotation files to the design. For more information on how SDF is used in the design, see Specifying SDF Files for Simulation.

STEP 4 - Simulate the Design
Once you have successfully loaded the design, simulation time is set to zero, and you must enter a run command to begin simulation. For more information, see Virology and System Verilog Simulation, and VHDL Simulation.

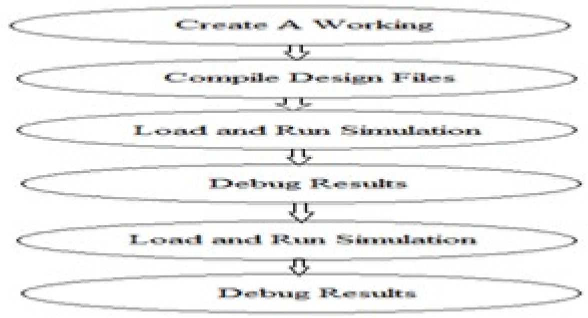

Fig.11: Simulation Flow

XI LINX ISE

Xilinx is disclosing this document and intellectual property (hereinafter “the design”) to you for use in the development of designs to operate on, or interface with Xilinx FPGAs. except as stated herein, none of the design may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Any unauthorized use of the design may violate copyright laws, trademark laws, the laws of privacy and publicity, and communications regulations and statutes.

Xilinx does not assume any liability arising out of the application or use of the design; nor does Xilinx convey any license under its patents, copyrights, or any rights of others. You are responsible for obtaining any rights you may require for your use or implementation of the design. Xilinx reserves the right to make changes, at any time, to the design as deemed desirable in the sole discretion of Xilinx. Xilinx assumes no obligation to correct any errors contained herein or to advise you of any

IMPLEMENTING THE DESIGN

- Select the counter source file in the Sources window.
- Open the Design Summary by double-clicking the View Design Summary process in the Processes tab.
- Double-click the Implement Design process in the Processes tab.
- Notice that after Implementation is complete, the Implementation processes have a green check mark next to them indicating that they completed successfully without Errors or Warnings.
- Locate the Performance Summary table near the bottom of the Design Summary.
- Click the All Constraints Met link in the Timing Constraints field to view the timing constraints report. Verify that the design meets the specified timing requirements.
- Close the Design Summary.

CONCLUSION

The proposed method of a built-in self-test (BIST) design for fault detection and fault diagnosis of static-RAM (SRAM)-based field-programmable gate arrays

(FPGAs). The target fault detection/diagnosis of the proposed BIST structure are open/short and delay faults in the wire channels, stuck on/off faults in PSs, andstuck-at-0/1 faults in LUTs. Low power scan based bits techniques have been studied along with literature papers. BIST based fault techniques are identified for SRAM based FPGA. Two fault models are designed and simulation outputs are discussed. Further fault models will discuss in Phase 2.

REFERENCES:

1. Abu-Issa A.S. and Quigley S.F. , (May 2009) "Bit-swapping LFSR and scan-chain ordering: A novel technique for peak- and average-power reduction in scan-based BIST," IEEE Trans. Compute.-Aided Des. Integer. Circuits Syst., vol. 28, no. 5, pp. 755–759.
2. Agrawal V.D. , Kim C.R, and K. K. SaludaK.K, (Mar.1993) "A tutorial on built-in self test. I. Principles," IEEE Des. Test Compute., vol. 10, no. 1, pp. 73–82.
3. Al-Yamani A., Devta-Prasanna N, Chelae E, Grinch M, Ana. Gonad, (May.2009) "Scan test cost and power reduction through systematic scan reconfiguration," IEEE Trans. Compute.-Aided Des. Integer. Circuits Syst.,vol. 26, no. 5, pp. 907–918.
4. Synopsys. ASTRO: Advanced Place-and-Route Solution for SoC Design, accessed on Mar. 1, 2015. [Online]. Available:http://www.synopsys.com/products/astro/astro.html
5. Banerjee S, Chowdhury D.R , and Bhattacharya B.B, (Jul. 2007) "An efficient scan tree design for compact test pattern set," IEEE Trans. Compute.-Aided Des. Integer. Circuits Syst., vol. 26, no. 7, pp. 1331–1339
6. Bardwell P.H , Caney W.H , and Saver J, Built in Test for VLSI: Pseudorandom Techniques. New York, NY, USA: Wiley, 1987.
7. Basturkmen N.Z, Reddy S.M., and I. Pomeranz, (Dec.2003) "A low power pseudorandom BIST technique," J. Electron. Test., Theory Appl., vol. 19, no. 6,pp. 637–644.
8. Bushnell M.L. and .Agrawal V..D., Essentials of Electronic Testing. Norwell, MA, USA: Clawer, 2000.