RESEARCH ARTICLE OPEN ACCESS

Optimization of Test Pattern Using Genetic Algorithm for Testing SRAM

Lekashri S
PhD Scholar, Anna University, Chennai, Tamil Nadu, India

ABSTRACT
An optimization of test pattern for testing of a Static Random Access Memory (SRAM) using genetic algorithm interconnects presented here is a method that associates a turn on inputs to numerous nets, which gives rise to test vectors to determine stuck-at, open, and bridging faults. This set up gives us privilege in reducing unnecessary composition that reduces the testing time for application-dependent testing for coverage of faults. This optimized test pattern is used as a test source for testing a circuit and identifying the faults in the circuit. The faults which are covered in are stuck at open and bridging faults. Genetic algorithm reduces the redundancy and optimizes the test pattern which results in reduced testing time and power consumption.

Keywords: Test pattern generator (TPG), Genetic Algorithm (GA), Linear feedback shift register (LFSR) Build in self-test (BIST), Circuit under test (CUT), Output response analyzer (ORA).

I. INTRODUCTION

Built in Self-Test (BIST) is an emerging technique for testing complex VLSI systems. To test a design by using a BIST methodology, the design has to be modified (enhanced) in such a way that part of the circuit is used to test the design itself. Therefore, BIST is defined as a Design for testability (DFT) technique in which testing is accomplished through built-in hardware components. A general BIST is shown. It consists of a test source block, the Circuit under test (CUT), a test response analysis block and a test controller block, which manages the application of the tests. In a classical BIST scheme, the test source consists of a special kind of register, test pattern generator (TPG), which generates on-chip test patterns. Recently, a new hybrid BIST approach has been proposed. It enhances the design with a read only memory (ROM) for storing some deterministic test patterns. These stored test patterns are used to capture faults that cannot be detected by the test patterns generated by the on-chip TPG.

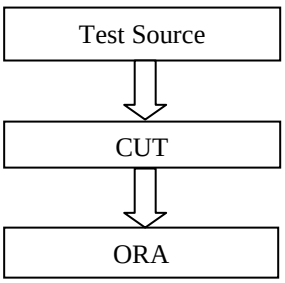


Figure 1: BIST Architecture

II. EXISTING MODEL

Application-dependent testing of a SRAM-based FPGA interconnect is been proposed. The novelty of this comprehensive method is that it connects an activating input to multiple nets, thus generating a compact set of activating test vectors and requiring a reduced numbers of configurations. The faults covered in this technique include all possible stuck-at, open, and pair wise bridging faults. Detection is not based on physical information (such as layout) of the FPGA interconnect; in this latter case, the possible adjacencies could be found and the number of pair wise bridging faults could be reduced. However, to allow a fair comparison with existing works logic simulation is therefore employed also in this paper; therefore, all possible stuck-at, open, and pair wise bridging faults are considered and detected by the proposed approach. So, for detecting the faults at the primary outputs, the induction fault detection method presented in is adopted. This is lower than a previous comprehensive method as well as by combining different methods. The existing method has a computational algorithm execution with L is the number of LUTs in the design. The activating input vectors required to sensitize the faults in the nets are generated using the Walsh code. The Walsh code for an interconnect with N nets is generated by its binary representation as a number. An activating input vector and the corresponding single-term function for a LUT for a configuration are derived from a set of activating inputs. The following block diagram describes about the activating input assignment,

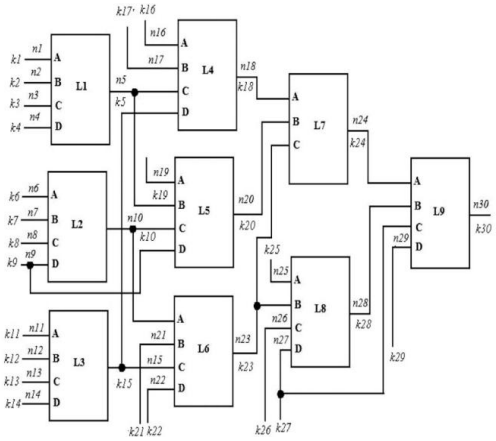


Figure 2: An Input Assignment

III. CONFIGURATION GENERATION ALGORITTHM

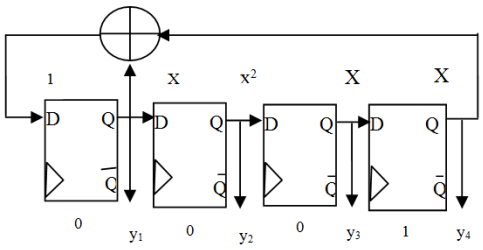
The process of assigning activating inputs and test vectors in the configurations for the nets starts by extracting the interconnect features of a specific circuit mapped to an FGPA (in this case, this is given by the Vertex 4) from the Native Generic Database (NGD) of the design. The NGD is converted to a text file and then the nets are sorted based on the number of connected LUTs. A greedy criterion for selecting the activating inputs and test vectors is used as follows: The first activating input is assigned to a net that is connected to the largest number of LUTs, followed by assigning the second and third activating inputs to the other nets in a descending order. However, during each activating input assignment, it must be ensured that no two (or more) nets connected to a LUT are driven by the same activating input. In the case of two nets connected to the same LUT having the same activating input, one of the activating inputs is assigned to the next available activating input to resolve the conflict. The pseudo code for the above process as given in the induction method of Tahoori [3] is utilized for propagating the sensitized faults to the primary output, thus accomplishing observability in detection.

At completion of the algorithm, the activating inputs to the LUTs are found and hence, the corresponding test configurations can be determined.

IV. PROPOSED METHOD

Optimization of test patterns for testing SRAM using genetic algorithm is been proposed. The novelty of this comprehensive method is that it connects an activating input to multiple nets, thus generating a compact set of activating test vectors and requiring a reduced numbers of configurations. The faults covered in this technique include all possible stuck-at, open, and pair wise bridging faults. However, to allow a fair comparison with existing works logic simulation is therefore employed also in this paper; therefore, all possible stuck-at, open, and pair wise bridging faults are considered and detected by the proposed approach. So, for detecting the faults at the primary outputs, the induction fault detection method is adopted. The proposed method optimizes the test pattern in execution .This is lower than a previous comprehensive method. The proposed method has a genetic algorithm execution. Therefore, the proposed method differs from previous approaches with respect to test pattern. The test patterns are generated by using Linear feedback shift register (LFSR).

An LFSR is a shift register with feedbacks from the last stage and other stages. The outputs of its flip-flops form the test pattern. Each state of the LFSR corresponds to one test pattern. The number of unique test patterns the LFSR can generate depends on the number and location of the feedbacks as well as its initial value, which is known as the seed. An example of an LFSR is shown in Figure 3



0	0	0	1
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
1	0	1	1
0	1	0	0
1	0	1	1
1	1	0	0
0	1	1	0
0	0	1	1
1	0	0	1
0	1	0	0
0	0	1	0
0	0	0	1

Figure 3: LFSR Example

It is initialized with the seed 0001. In the subsequent clock cycles, a series of test patterns are produced at the outputs of the flip-flops. This LFSR, which has n=4 flip-flops, produces a total of 15 (2n - 1) distinct patterns (except 0000) as shown in Figure.3. The feedback positions are usually described by a characteristic polynomial. In our example, feedbacks are made from the first (x) and the fourth (x4) positions, hence the characteristic polynomial of the LFSR is p(x) = 1 + x + x4. The

choice of feedback positions (the choice of the polynomial) determines the length of the test sequences generated. Special polynomials known as primitive polynomials give maximal length sequences (2n-1). A polynomial p(x) = 1 + x + x4, which is used in our example, is primitive. It generates a sequence of 15 distinct test patterns before repetition. Therefore, when designing an LFSR a good choice of seed and polynomial is crucial for generating a good sequence of tests. These test patterns optimized by using genetic algorithm (GA)

V. GENETIC ALGORITHM

Genetic algorithm (GA) is an adaptive heuristic search algorithm based on the mechanism of natural selection and evaluation. GA is an artificial intelligence procedure and robust search method. This technique is efficient for finding combinatorial optimization problem. The objective of GA is to find optimal solution to a problem. Genetic algorithm belongs to the class of evolutionary algorithm which generates solution to optimization problems using techniques inspired by natural evolution such as inheritance, mutation, selection and crossover.

VI. BLOCK DESCRIPTION

Proposed method Hardware Architecture of GA is shown in fig 4. Here the evolvable hardware is used. This evolvable hardware can be implemented by combining hardware architecture of GA with evolvable computing logic. This paper describes the implementation of evolvable hardware with the state machine hardware. The hardware architecture of genetic algorithm model based on FPGA consists of two units. They are processing unit and control unit.

Processing unit:

The function of the processing unit includes initial population generation, fitness evaluation and genetic operation. There are five hardware modules in the processing unit. They are generation modules, selection modules, crossover modules, mutation modules and random number generation module (RNG). RNG generates random number for other modules.

Control unit:

The control unit acts as a control state machine. The state machine of the control unit can be used to decide the operating sequence of initial population generation, population storage, fitness evaluation, selection, crossover and mutation. It can automatically send control signal to the processing unit.

Operation:

The control signal can assure a correct executing in circles of these modules in the processing unit, depending on the operating rule about the sequence of these operations. The control unit receives the current state signals and generates the next state. These two units work coordinately to perform the calculation of GA.

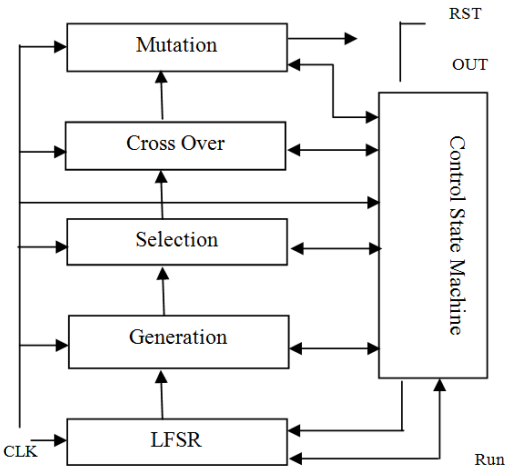


Figure 4: Hardware Architecture of GA

Control state machine:

The modules of processing unit are controlled by the control state machine of the control unit and can work on two states. They are active state and sleeping state. The figure 5 shows the binary decision diagram of the control state machine. The state machine consists of four states. They are idle, birth, GA, store.

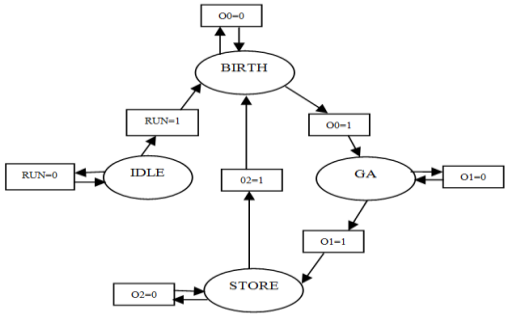


Figure 5: Control State Machine

The test generator starts with a random population of n individuals, and a (fault) simulator is used to calculate the fitness of each individual. The best test vector evolved in any generation is selected and added to the test set. Then, the fault set is updated by removing the detected faults by the added vectors. The GA process repeats itself until no more faults can be detected

VII. RESULTS AND DISCUSSION

Hence the genetic algorithm used here has avoided the redundancy and reduced the number of test patterns by optimization technique. It also reduced the time constraint with just ten Test Patterns. Only these optimized ten test patterns are

used for testing and fault analysis of the circuit. The simulated output of without fault and with fault are shown below in fig 6 & fig 7 respectively. Thus the observed results from the proposed architecture achieves minimum power in the range of 220-230 mW and the reduced time constraint is about 1.5-2 ns.

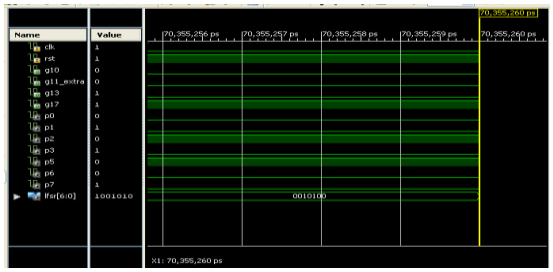


Figure 6: Without Fault

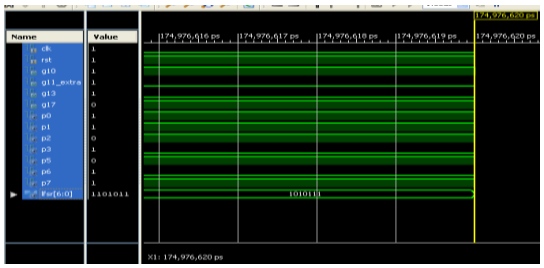


Figure 7: With Fault

VIII. CONCLUSION

Thus genetic algorithmic method for optimizing test pattern for testing of a SRAM that is been proposed relies on generating minimized test patterns. This algorithmic-based method detects all stuck-at, open, and pair wise bridging faults, optimizes test patterns with minimum time constraint and less power consumption.

REFERENCES

[1] M. J. O'Dare and T. Arslan,"Generating Test Patterns For Vlsi Circuits Using A Genetic Algorithm" ,U.K.

[2] Lalit A. Patel, Sarman K. Hadia,"Transistor Level Fault Finding in VLSI Circuits using Genetic Algorithm "

[3] M.B. Tahoori, "Application-Dependent Testing of FPGAs," IEEE Trans. Very Large Scale Integration Systems, vol. 14, no. 9, pp. 1024- 1033, Sept. 2006.

[4] M.B. Tahoori, "Application-Dependent Testing of FPGA Interconnects," Proc. 18th IEEE Int'l Symp. Defect and Fault Tolerance in VLSI Systems, pp. 409-416, Nov. 2003.

[5] M.B. Tahoori, "Application-Dependent Diagnosis of FPGAs," Proc. IEEE Int'l Test Conf., pp. 645-654, Oct. 2004.

[6] A. Doumar and H. Ito, "Testing the Logic Cells and Interconnect Resources for FPGAs," Proc. Eighth Asian Test Symp., pp. 369-374, Nov. 1999.

[7] Y. Yu, J. Xu, W.K. Huang, and F. Lombardi, "A Diagnosis Method for Interconnects in SRAM Based FPGAs," Proc. Seventh Asian Test Symp., pp. 278-282, Dec. 1998.

[8] W.K. Huang, X.T. Chen, and F. Lombardi, "On the Diagnosis of Programmable Interconnect System: Theory and Application," Proc. 14th VLSl Test Symp., pp. 204-209, Apr. 1996.

[9] F. Lombardi, D. Ashen, X. Chen, and W.K. Huang, "Diagnosing Programmable Interconnect System for FPGAs," Proc. Fourth ACM Int'l Symp. Field-Programmable Gate Arrays, pp. 100-106, 1996.

[10] Feng Liang, Luwen Zhang, Shaochong Lei, Guohe Zhang, Kaile Gao an