LOW POWER AND AREA EFFICIENT 3 – WEIGHT PATTERN GENERATION USING ACCUMULATOR

P.Gopi[#1], L. Ramamurthy[*2], C.Chandrasekhar[#3] P.Anilkumar[*2]

[#]*M.Tech Student, Department of ECE & Vemu institute of Technology*

P.Kothakota, Near pakala, Chittoor, A.P, India

Abstract — Current VLSI circuits, e.g., data path architectures, or digital signal processing chips common only contain arithmetic modules [accumulators or arithmetic logic units (ALUs)].This has fired the idea if arithmetic BIST (ABIST).The basic idea of ABIST is to utilize accumulators for built-in testing (compression of the CUT responses, or generation of test patterns) and has been shown to result in low hardware overhead and low impact on the circuit normal operating speed. An accumulator based test pattern generation scheme that compares favourably to previously proposed schemes .it was proved that the test vectors generated by an accumulator whose input share driven by a constant pattern can have acceptable pseudorandom characteristics, If the input pattern is properly selected. However, modules containing hard-to-detect faults still require extra test hardware either by inserting test points into them mission logic or by storing additional deterministic test patterns.

Weighted pseudorandom built-in self test (BIST) schemes have been utilized in order to drive down the number of vectors to achieve complete fault coverage in BIST applications. Weighted sets comprising three weights, namely 0, 1, and 0.5 have been successfully utilized so far for test pattern generation, since they result in both low testing time and low consumed power. Here accumulator-based 3-weight test pattern generation scheme is presented; the proposed scheme generates set of patterns with weights 0, 0.5,1. Since accumulators are commonly found in current VLSI chips, this scheme can be efficiently utilized to drive down the hardware of BIST pattern generation, as well. .

Keywords — *accumulator, built in self-test, vectors, pseudorandom, fault detection.*

I. INTRODUCTION

Pseudo random test generation for combinational circuits was proposed to reduce the cost of storing tests for Built-In-Test. Pseudo-random patterns can easily be generated by Linear Feedback Shift Registers(LFSR's) or Cellular Automata , whose area requirement scan be smaller than the size of a RAM (or a counter/decoder structure) required for generating deterministic tests. In addition, application of moderate numbers of pseudo-random tests enhances the coverage of un modelled faults. Since it was discovered that some circuits contain faults for which large numbers of random patterns have to be generated before a high fault coverage can be reached, random patterns are biased , by changing the probability of obtaining a 0 or a 1 on a given input, from half (for pure pseudo-random tests) to some other value. A set of weight assignments is selected to allow high fault coverage to be obtained by random pattern sequences of practical length (a weight assignment includes a weight for every primary input of the circuit).

Extensive work has been done on the computation of a small number of weight assignments for random testing of combinational circuits, and different approaches for the computation of weights have been proposed. Weighted random pattern methods which rely on a single weight assignment, usually fail to achieve complete fault coverage using a reasonable number of test patterns. The reason for this is that the weights are computed to be suitable for most faults,

ISSN: 2278 – 7798

but some faults may require long test sequences to be detected with weights that do not match their activation and propagation requirements.

A compromise between fault coverage and the number of test patterns is made, which in most cases results in incomplete fault coverage. Multiple weight assignments were, therefore, suggested in cases where different faults require different biases of the input combinations applied to the circuit, to ensure that a small number of random patterns detect all faults, multiple weight assignments that are computed based on fault detection probabilities. The number of weight assignments varies from 1 to 6 for ISCAS-85 circuit. Approaches to derive weight assignments which are based on given deterministic test sets are especially attractive since they have the potential to allow complete coverage to be obtained by small number of random patterns. Weights are computed based on the distribution of 1's and 0's in a deterministic test set however, at most two weight assignments are used, and incomplete fault coverage is reported in several cases. The correlations between the values assigned to different primary inputs by a deterministic test set are analyzed, and random patterns are generated based on the computed correlations. All weighted random and related methods suffer from the following limitation.

To produce weights which are different from 0.5, several cells of an LFSR or a shift register are connected to a gate whose output is used to derive the corresponding primary input of the circuit under test; e.g., to produce a weight of 0.25, two cells of the LFSR are connected to an AND gate, whose output drives a primary input of the circuit. When weights are allowed to assume arbitrary values, arbitrary numbers of shift-register cells have to be used to produce the required input values. Register cells are generally not allowed to be shared between circuits that generate weights for different primary inputs, to avoid correlation between the values different primary inputs assume. In experimental evidence was presented to demonstrate that in some cases, such correlations may not affect the fault coverage Achieved. However, some faults remain undetected in many of the circuits considered.

A different approach to efficient generation of tests is reported in where special hardware is used to produce a subset of a deterministic test set, and detection of all faults is based on generating the selected subset of deterministic tests as part of a larger set of input combinations which can be regarded as random patterns for faults not detected by the target subset of tests, The hardware consists of a counter and XOR gates.

In some cases, the hardware cost to achieve complete fault coverage may become high, or the number of test patterns may become extremely large. Work on efficient generation of tests thus varies from pure random tests, which have low hardware requirements but may require large test sequences (or, alternatively, may leave some faults undetected), to application of a deterministic test set, which is short but may have a high hardware cost and may not provide adequate coverage of un modelled faults. Other approaches that were investigated, and which mix deterministic tests and random tests, include the use of several seeds, or initial LFSR states, to generate random patterns , instead of a single seed conventionally used, and the use of transformations, other than the one performed by the LFSR, on the states of the LFSR Random pattern generation for sequential circuits was investigated in this project, we propose a method that dynamically walks through the range of possible test generation approaches, starting from pure random tests to detect easy to-detect faults at low hardware cost, then reducing the number of inputs which are allowed to be specified randomly, fixing an increasing number of the inputs to 0 or 1 according to a given deterministic test set, to detect faults which have more stringent requirements on input values and cannot be detected by a purely random sequence of reasonable length, finally allowing deterministic tests to be applied for faults that could not be detected otherwise (if such faults remain). The method can thus be viewed as a weighted random test generation method that uses three weights: a weight of 0.5 indicates pure random selection of values; a

ISSN: 2278 – 7798

weight of 0 corresponds to fixing an input to 0; and a weight of 1 corresponds to fixing an input to 1. The first assignment of weights assigns a weight of 0.5 to all primary inputs, resulting in pure random tests, while succeeding weight assignments are closer to a given deterministic test set, having increasing numbers of 0 and 1 weights and decreasing numbers of 0.5 weights. A minimal number of weight assignments is searched for, to keep hardware requirements low, and at the same time, the number of tests generated for every weight assignment is limited, to limit the total test length. In contrast to other weighted random methods, a fixed number of random tests (sufficient to detect all target faults) is generated for every weight assignment to avoid the hardware overhead related to the use of different numbers of tests.

II. TYPES OF PSEUDO RANDOM SCEMES

1. LFSR
2. Cellular Automata
3. Accumulator

2.1 *LFSR* (Linear Feedback Shift Register)

Pseudorandom testing is based on testing a circuit with test patterns that have the characteristics of random patterns but where the patterns are generated deterministically and are, therefore, repeatable. These patterns may be generated with or without replacement. Replacement implies that a pattern is repeated. As opposed to exhaustive testing, all 2n patterns do not need to be generated. This characteristic alone reduces the overhead as compared to exhaustive testing. Pseudorandom testing will introduce a very important type of TPG circuit, the Linear Feedback Shift Register (LFSR) work shown in Figure: 1.1. On the most basic level these are D flip-flops with feedback. The feedback scheme determines what value is fed back through the registers. This scheme is based on the insertion of XOR gates at specific points in the feedback loop.The placement of these gates can be described mathematically with primitive polynomials and therefore deterministically placed to generate various states.

There are many different ways to implement an LFSR, depending on which patterns one wishes to generate. An LFSR that produces the maximum number of patterns for a specified bit length is appropriately named a maximal length LFSR. This is very similar to a counter but the patterns will not be incrementally generated and the all-zeros pattern will not appear (often an important test).

The all-zeros pattern cannot be generated in an LFSR if the LSFR is to continue to generate patterns because of its feedback and XOR nature. However, with external logic (and increased overhead) the all zeros pattern can be incorporated. The major drawback to pseudorandom testing is that some circuits contain random pattern resistant faults and thus require long test lengths to insure high fault coverage.

Also inherent in LFSRs is the tendency to produce patterns having equal numbers of 0s and 1s on each output line. For some circuits it is better to bias the distribution of 0s and 1s to achieve higher fault coverage. Overall designs based strictly on pseudorandom test pattern generation are sufficient only in very basic applications. The TPGs for these setups create low overhead implementations but the fault coverage is often unacceptable.

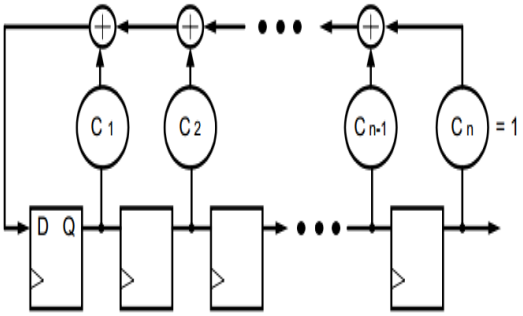


Figure 1.1: General Structure of LFSR

Definition: If period p of sequence generated by an LFSR is 2n -1, then it is a maximum length sequence.

Definition: The characteristic polynomial associated with a maximum length sequence is a primitive polynomial.

With pseudorandom BIST theory in place, the goal becomes to produce designs that move away from that end of the BIST spectrum, toward the middle to produce better coverage. To demonstrate this, Circular Self-Test Path (CSTP) will be examined. CSTP is an excellent example of a low

ISSN: 2278 – 7798

overhead design with questionable (but often improved) fault coverage based on design techniques from the pseudorandom ideology.

The Circular Self-Test Path technique is linking selected registers of the circuit into one long circular register, creating a feedback shift register. This also provides a data compaction capability by augmenting each cell of the circular path with an extra gate to XOR its functional input with the output of the preceding cell, as shown in Figure: 1.2 Once this is achieved, the registers have three modes of operation: normal, test, and scan.

In normal mode all registers that make up the circular path serve as D-type memory elements, with the combinational logic (CL) blocks in the circuit being fed as normal. In test mode the modified registers form a circular path as shown in Figure: 1.3. After resetting the circuit, the circular path provides the CL blocks with test patterns and simultaneously, the CL responses are compacted.

A decision regarding the result of the testing is made by comparing the stream of bits available at the output of the circuit (driven by one of the circular output registers) with its fault free value. Within this mode the similarities between CSTP and pseudorandom generation become clearer. Both utilize shift registers to run patterns through the CUT.

However, the patterns run through CUT in CSTP are neither pseudorandom nor random. They are dependent on the structure and more importantly the functionality of the CUT. This phenomenon will be discussed shortly. The final mode of operation is scan, which is needed if a global reset is not available. In this case, the scan mode will allow circuit initialization to occur by serially scanning values through the chain. This will allow further manipulation of the test environment.

Several observations ensue from this description of how CSTP functions. First, only a single test session is required, minimizing the complexity of the controller and potentially reducing the test running time. Second, this is an excellent example of low overhead (the reason why this is discussed at this point in the paper). The only additional overhead lies in the modification of the current registers along the boundary to CSTP registers.

A cell of the circular path implements only two basic functions: parallel load (for normal operation) and data compaction (for test mode). The third mode (scan) is needed only if no reset is present. Unlike other multifunctional registers, a CSTP register does not require any logic to implement the local feedback. This leads to very little overhead in terms of size added to the circuit.

So, CTSP (like most BIST schemes of this nature) has low overhead, but what are the downsides.As mentioned earlier, the patterns generated by this method are determined by the structure of the CL, they are not random or pseudorandom. Therefore, certain structures may not lend themselves well to these types of tests. Also testing efficiency estimates for these methods cannot be applied. It is for this reason that the fault coverage remains suspect.

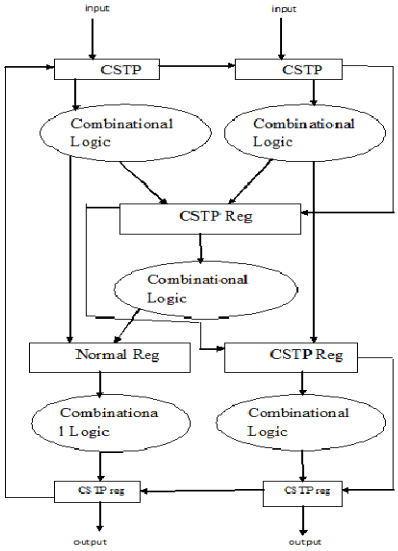


Figure 1.2: Block diagram for circular self testing path.

2.2 *CELLULAR AUTOMATA*

Cellular Automata (CA) are simply grids of cells, where the individual cells change states according to a set of rules. The CA may be one dimensional, or linear, like a string of cells in a row (below), or two dimensional, like a checkerboard.

ISSN: 2278 – 7798

A cellular automata consists of a regular grid of cells, each in one of a finite number of states, such as on and off (in contrast to a coupled map lattice). The grid can be in any finite number of dimensions. For each cell, a set of cells called its neighborhood is defined relative to the specified cell. An initial state (time t=0) is selected by assigning a state for each cell. A new generation is created (advancing t by 1), according to some fixed rule (generally, a mathematical function) that determines the new state of each cell in terms of the current state of the cell and the states of the cells in its neighbourhood. Typically, the rule for updating the state of cells is the same for each cell and does not change over time, and is applied to the whole grid simultaneously, though exceptions are known, such as the stochastic cellular automaton and asynchronous cellular automaton.

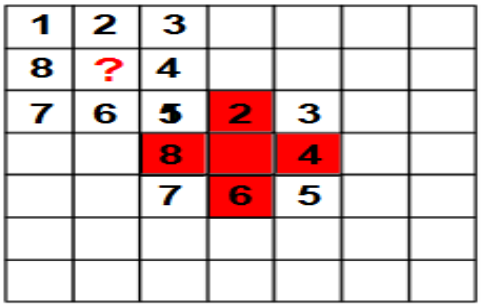


Figure 1.3: Diagram for cellular Automata

Rules:

- Survival Rules – a live cell survives to the next generation if at least 2 but no more than three of the surrounding 8 cells are alive. Less than 2 and it dies of loneliness; more than 3 and it dies of overcrowding.
- Birth Rules – a dead cells comes alive the next generation if 3, any 3, of the surrounding 8 cells are also alive

2.3 *ACCUMULATOR*

In a computer's central processing unit (CPU), an accumulator is a register in which intermediate arithmetic and logic results are stored.Without a register like an accumulator, it would be necessary to write the result of each calculation (addition, multiplication, shift, etc.) to main memory, perhaps only to be read right back again for use in the next operation. Access to main memory is slower than access to a register like the accumulator because the technology used for the large main memory is slower (but cheaper) than that used for a register. Early electronic computer systems were often split into two groups, those with accumulators and those without.

III. ADDERS, PATTERN GENERATION AND FAULT DETECTION

3.1 W-ADDER DESIGN

The implementation of AWPG is based on an accumulator comprising a register whose inputs are driven by a binary adder; the one input of the adder is the input of the accumulator whilst its other input is the output of the register . In order to implement AWPG, the adder of the accumulator is modified as presented in Figure: 1.4(b). We shall refer to the design of the modified adder with the term W- adder. In the sequel we shall present the modification on the adder for the case of the ripple-carry adder.

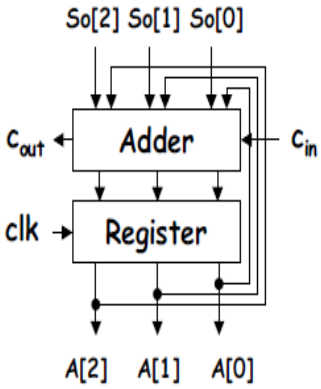


Figure 1.4: (a) Accumulator

ISSN: 2278 – 7798

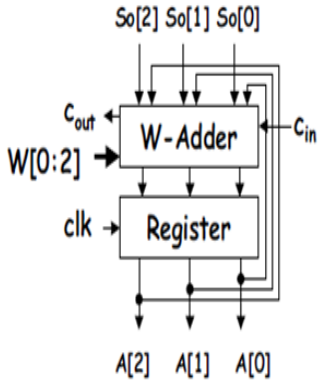


Figure 1.4: (b) Accumulator for AWPG

The ripple carry implementation of the binary adder is presented in Figure: 1.5(a). In Figure: 1.5(b) a NAND-XOR implementation of the Full Adder cell is presented.

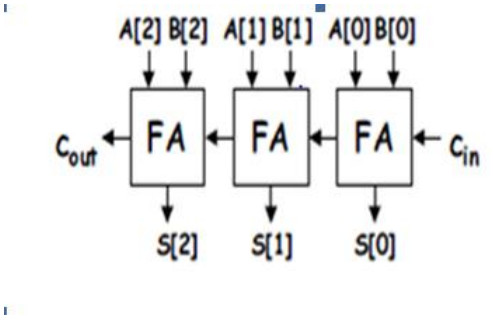


Figure 1.5: (a) Ripple Carry Adder

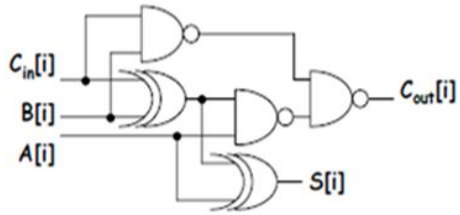


Figure 1.5: (b) Full Adder Cell

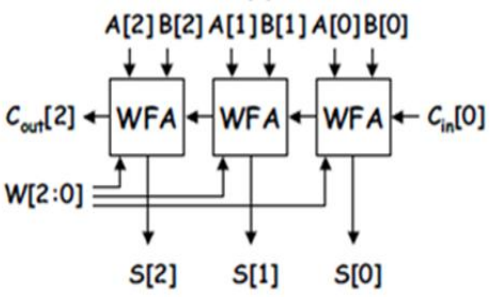


Figure 1.6: Ripple carry w-adder

The WFA cell takes an extra input, W[i]; when W[i] =0, the WFA cell operates identically to the full adder cell. When W[i] =1, cin does not affect the S operation, and S=A, while cout = cin. The operation of the WFA cell is presented in Figure: 1.6.

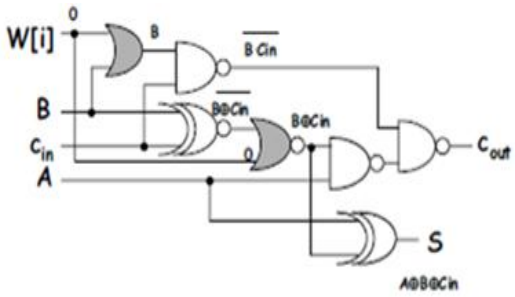


Figure 1.7: (a) Operation of the w-adder cell at w=0

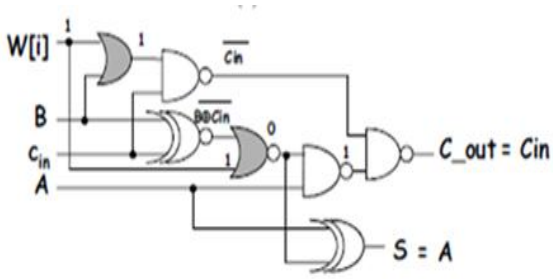


Figure 1.7: (b) Operation of the w-adder cell at w=1

3.2 ACCUMULATOR-BASED 3-WEIGHT PATTERN GENERATION:

Starting from this deterministic test set, in order to apply the3-weight pattern generation scheme, one of the schemes proposed in previous schemes can be utilized. According to these schemes, a typical weight assignment procedure would involve separating the test set into two subsets, S1 and S2 as follows:

In the first assignment, inputs A[2] and A[0] are constantly driven by “1”, while inputs A[4], A[3], A[1] are pseudo randomly generated (i.e., have weights 0.5). Similarly, in the second weight assignment (subset S2), inputs A[2] and A[0] are constantly driven by “0”, input A[1] is driven by “1” and inputs A[4] and A[3] are pseudo randomly generated.

The above reasoning calls for a configuration of the accumulator, where the following conditions are met: 1) an accumulator output can be constantly driven by “1” or “0” and

ISSN: 2278 – 7798

2) an accumulator cell with its output constantly driven to "1" or "0" allows the carry input of the stage to transfer to its carry output unchanged. This latter condition is required in order to effectively generate pseudorandom patterns in the accumulator outputs whose weight assignment is "_".

3.3 DESIGN METHODOLOGY

The implementation of the weighted-pattern generation scheme is based on the full adder truth table, presented in Table III. From Table III we can see that in lines #2, #3, #6, and #7 of the truth table, Therefore, in order to transfer the carry input to the carry output, it is enough to set .The proposed scheme is based on this observation.

The implementation of the proposed weighted pattern generation scheme is based on the accumulator cell presented in Figure: 1.10, which consists of a Full Adder (FA) cell and a D-type flip-flop with asynchronous set and reset inputs whose output is also driven to one of the full adder inputs. In Figure: 1.11, we assume, without loss of generality, that the set and reset are active high signals. In the same figure the respective cell of the driving register B[i] is also shown. For this accumulator cell, one out of three configurations can be utilized, as shown in Figure: 1.11.

#	Cin	A[i]	B[i]	Cout	Comment
1	0	0	0	0	
2	0	0	1	0	Cout= Cin
3	0	1	1	0	Cout= Cin
4	0	1	0	1	
5	1	0	1	0	
6	1	0	0	1	Cout= Cin
7	1	1	0	1	Cout= Cin
8	1	1	1	1	

Table 1: Truth table of Full adder

In Figure: 1.11(c), we present the configuration that drives the CUT inputs when A[i] = "_" is required. Set[i]=0 and Reset[i]=0 The D input of the flip-flop of register B is driven by either 1 or 0, depending on the value that will be added to the accumulator inputs in order to generate satisfactorily random patterns to the inputs of the CUT. In Fig.1.12, the general configuration of the existing scheme is presented.

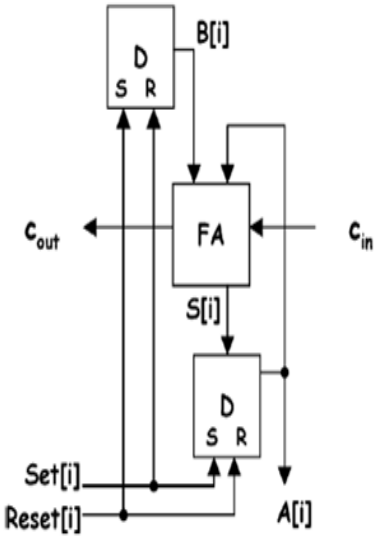
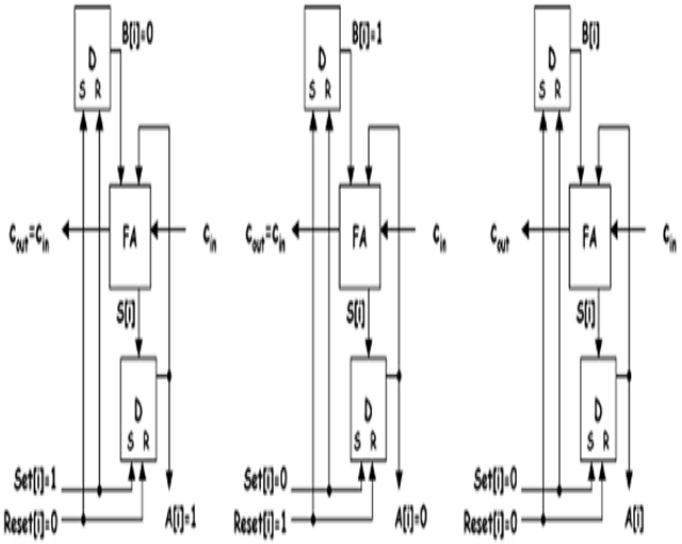


Figure 1.10: Accumulator Cell



(a) (b) (c)

Figure 1.11: Configuration of accumulator cell

The Logic module provides the Set [n-1:0] and Reset [n-1:0] signals that drive the S and R inputs of the register A and Register B inputs. Note that the signals that drive the S inputs of the flip-flops of Register A, also drive the R inputs of the flip-flops of Register B and vice versa.

ISSN: 2278 – 7798

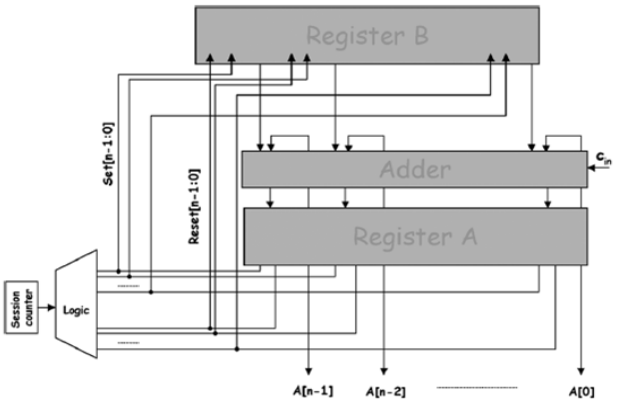


Figure 1.12: Accumulator based pattern generation.

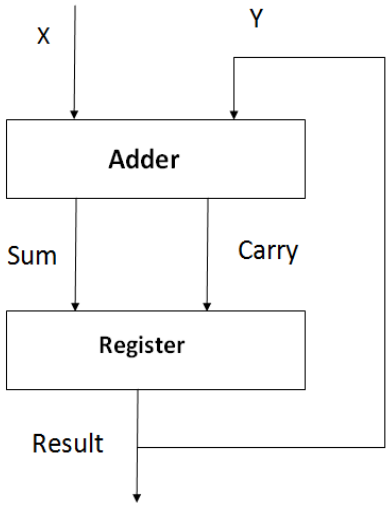


Figure 1.13: Flow of pattern generation

3.4 FAULT & FAULT MODELS

A fault is the representation of a defect reflecting a physical condition that causes a circuit to fail to perform in a required manner. A failure is a deviation in the performance of a circuit or system from its specified behaviour and represents an irreversible state of a component such that it must be repaired in order for it to provide its intended design function. A circuit error is a wrong output signal produced by a defective circuit. A circuit defect may lead to a fault, and it can result in a system failure.

The reduction in feature size increases the probability that a manufacturing defect in the IC will result in a faulty chip. FPGA's are no exception from this. A very small defect can easily found when the feature size of the device is less than 100 nm. Furthermore, it takes only one faulty CLB or wire to make the entire FPGA fail to function properly. Yet, defects created during the manufacturing process are unavoidable and, as a result, some number of FPGA's is expected to be faulty; therefore, testing is required to guarantee fault free FPGA's.

Faults can be divided into two categories:

1. Permanent Faults
2. Transient Faults

Fabrication faults and design faults are among the Permanent faults. Transient faults, commonly called single event upsets (SEUs), are brief incorrect values resulting from external forces (terrestrial radiation, particles from solar flares, cosmic rays, and radiation from other space phenomena) altering the balance or locations of electrons, usually in a small area of the system. Tolerating permanent faults is critical to maximizing device and system yields to decrease costs, and to increasing the lifespan of deployed devices.

To effectively evaluate the quality of a set of tests for a product, as well as to evaluate the effectiveness of a BIST approach in its application to that product, fault models are required for emulation of faults or defects in a simulation environment. Because of the diversity of defects, it is difficult to generate tests for real defects. Fault models are necessary for generating and evaluating a set of test vectors.

Fault models can be divided into three types

1. Interconnect fault model
2. Logic Block fault model
3. Delay Fault

Logic Block fault model: Logical faults represent the effect of physical faults on the behaviour of the system error more effectively when compared to LFSR based and LPLFSR based fault detection.

- The causes of FPGA delay fault
- More circuits operating at high speed
- DSM processes have resulted in more defects affecting the delay

ISSN: 2278 – 7798

FPGA delay testing problem

- Path delay fault model is not appropriate
  - Segment delay fault model is used
- At-speed testing is practically difficult
  - Need BIST approach

The circuit is modeled as an interconnection of Boolean gates each connecting line can have two types of faults

- Stuck-at-1 (s-a-1) & Stuck-at-0 (s-a-0) A circuit with n lines can have 3^n-1 possible stuck line combinations
- An n-line circuit can have at most 2n single stuck-at faults
- Three properties define a single stuck-at fault
  1. Only one line is faulty
  2. The faulty line is permanently set to 0 or 1
  3. The fault can be at an input or output of a gate
- MOS transistor is considered an ideal switch and two types of faults are modeled:
  1. Stuck-open -- a single transistor is permanently stuck in the open state.
  2. Stuck-short -- a single transistor is permanently shorted irrespective of its gate voltage.
- Detection of a stuck-open fault requires two vectors.

IV. SIMULATIONS RESULTS

4.1 ERROR OUTPUT DUE TO FAULT IN THE CIRCUT

Simulation results for fault detection are as shown in below.

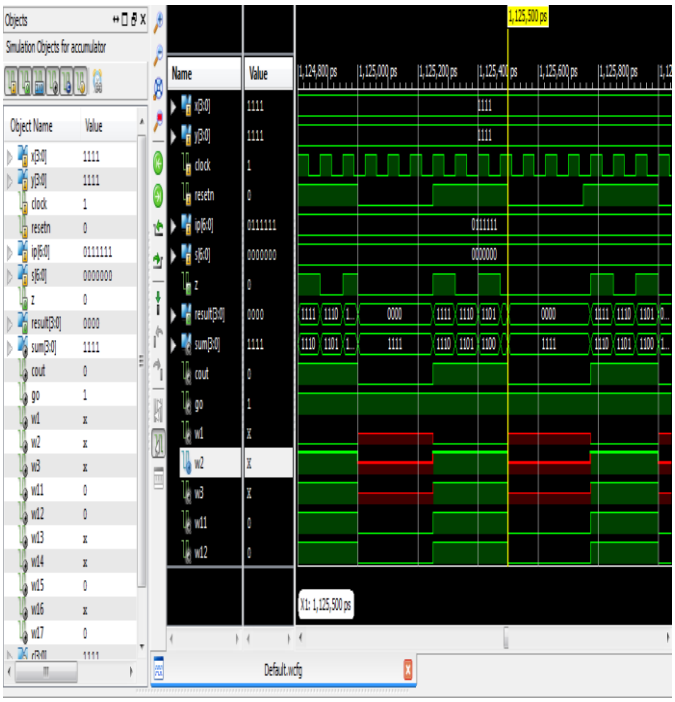
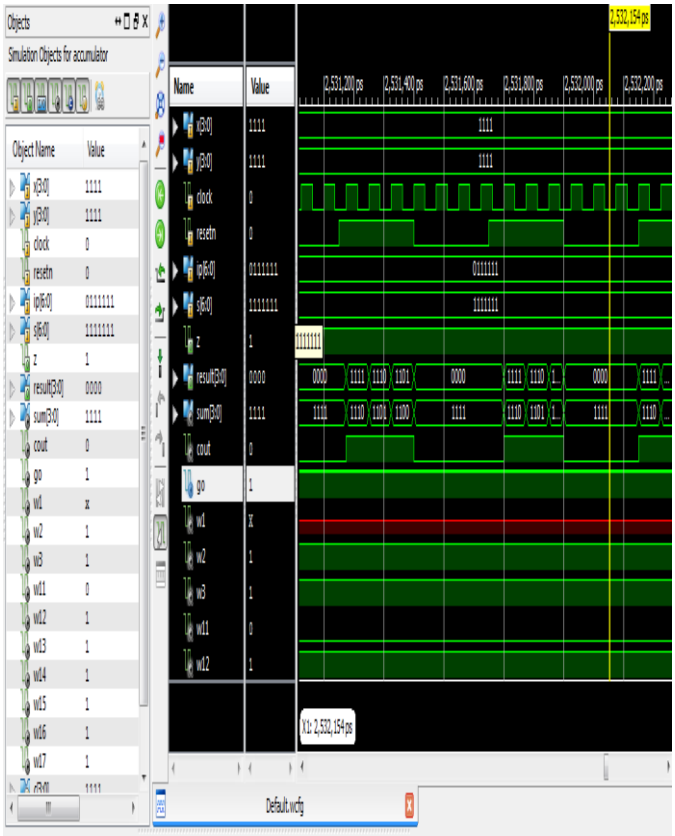


Figure 1.15: Simulation results for Fault detection



ISSN: 2278 – 7798

Figure 1.16: Fault detection

V. CONCLUSION

We have presented an accumulator-based 3-weight (0, 0.5, and 1)test-per-clock generation scheme, which can be utilized to efficiently generate weighted patterns without altering the structure of the adder. Comparisons with a previously proposed accumulator-based3-weight pattern generation technique indicate that the hardware overhead of the proposed scheme is lower (75%), while at the same time no redesign of the accumulator is imposed, thus resulting in reduction of 20%–95% in test application time. Comparisons with scan based schemes show that the proposed schemes results in lower hardware overhead. Finally, comparisons with the accumulator- based scheme proposed in reveal that the proposed scheme results in significant decrease (98%) in hardware overhead.

Acknowledgment

We sincerely thanks to L. Ramamurthy, Dean, Vemu institute of Technology, Mr. C. Chandrasekhar, Assoc. Professor, Mr. P. Anilkumar, Asst. Professor, Mother Theresa Institute of Engineering & Technology, Palamaner, Principal of Vemu institute of Technology, Management of VIT, and the Staff members of ECE Dept. VIT, family members, and friends, one and all who helped us to make this paper successful.

REFERENCES

[1] J. Breckling, Ed., *The Analysis of Directional Time Series: Applications to Wind Speed and Direction*, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.

[2] P. Hortensius , R. McLeod , W. Pries , M. Miller and H. Card "Cellular automata-based pseudorandom generators for built-in self test", IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 8, no. 8, pp.842 -859 .

[3] A. Stroele "A self test approach using accumulators as test pattern generators", Proc. Int. Symp. Circuits Syst., pp.2120 -2123 1995

[4] H. J. Wunderlich "Multiple distributions for biased random test patterns", Proc. IEEE Int. Test Conf., pp.236 -244 1988 .

[5] I. Pomeranz and S. M. Reddy "3 weight pseudo-random test generation based on a deterministic test set for combinational and sequential circuits", IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 12, no. 7, pp.1050 -1058

[6] K. Radecka , J. Rajski and J. Tyszer "Arithmetic built-in self-test for DSP cores", IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 16, no. 11, pp.1358 -13691997

ISSN: 2278 – 7798