REDUCED LATENCY OF 3 WEIGHT PATTERN GENERATION USING ARITHMETIC BIST

Siva Jyothi Bellamkonda¹ and M.Pardha Saradhi²

¹PG student Department of ECE, Vasireddy Venkatadri institute of Technology, Nambur, Guntur, Andhra Pradesh, India

²Associate professor Department of ECE, Vasireddy Venkatadri institute of Technology, Nambur, Guntur, Andhra Pradesh, India

Corresponding author E mail id: bsivajyothi5@gmail.com

Abstract

In this paper an accumulator-based 3-weight test pattern generation scheme is presented. The proposed scheme copes with the inherent drawbacks of the scheme proposed more precisely. First, it does not impose any requirements about the design of the adder i.e., it can be implemented using any adder design and then it does not require any modification of the adder; and hence it does not affect the operating speed of the adder. Furthermore, the proposed scheme compares favorably to the scheme proposed in terms of the required hardware overhead. The weighted random test pattern generation represents a significant departure from classical methods of generating test sequences for complex large scale integration packages. The virtue of this technique is its simplicity and the fact that test-generation time is virtually independent of or gates in the logic package to be tested. This technique can be used both in a conventional tester and in a tester where the weighted random test pattern generation is implemented in hardware. By this proposed method the patterns are generated with reduced delay.

1. INTRODUCTION

BIST is a design-for-testability technique that places the testing functions physically with the circuit under test (CUT). The basic BIST architecture requires the addition of three hardware blocks to a digital circuit: a test pattern generator, a response analyzer, and a test controller. The test pattern generator generates the test patterns for the CUT. Examples of pattern generators are a ROM with stored patterns, a counter, and a linear feedback shift register (LFSR). A typical response analyzer is a comparator with stored responses or an LFSR used as a signature analyzer. It compacts and analyzes the test responses to determine correctness of the CUT. A test control block is necessary to activate the test and analyze the responses. However, in general, several test-related functions can be executed through a test controller circuit.

A new weighted random pattern design for testability is described where the shift register latches distributed throughout the chip are modified so that they can generate biased pseudo-random patterns upon demand. A two-bit code is transmitted to each weighted random pattern shift register latches to determine its specific weight. The weighted random pattern test is then divided into groups, where each group is activated with a different set of weights. The weights are dynamically adjusted during the course of the test to "go after" the remaining untested faults.

An accumulator-based 3-weight test pattern generation scheme is presented; the proposed scheme generates set of patterns with weights 0, 0.5, and 1. Since accumulators are commonly found in current VLSI chips, this scheme can be efficiently utilized to drive down the hardware of built in self test pattern generation, as well. Comparisons with previously presented schemes indicate that the proposed scheme compares favorably with respect to the required hardware.

2. WEIGHT GENERATING COMBINATIONAL CIRCUIT

A digital system is tested and diagnosed during its lifetime on numerous occasions. Such a test and diagnosis should be quick and have very high fault coverage. One way to ensure this is to specify such a testing to as one of the system functions, so now it is called Built In Self Test (BIST). With properly designed BIST, the cost of added
test hardware will be more than balanced by the benefits in terms of reliability and reduced maintenance cost.

2.1 WEIGHTED GENERATION OF BIST

For BIST, we would require that the test patterns be generated on the system/chip itself. However, this should be done keeping in mind that the additional hardware is minimized. One extreme is to use exhaustive testing using a counter and storing the results for each fault simulation at a place on the chip (like ROM). An n input circuit would then require $2^n$ combinations which can be very tiresome on the system with respect to the space and the time. Also, more the number of transitions, the power consumed will be more.

2.1.1 BIST Techniques

The BIST techniques are classified based on the operational condition of the circuit Under test (CUT)

1. Off-Line BIST
2. On-Line BIST

ON-LINE BIST

Testing occurs during normal functional operating conditions (No test mode, Real-Time error detection).

1. Concurrent : Occurs simultaneously with normal functional operation (Realized by using coding techniques).
2. Non-concurrent : Carried out while in idle state (Interruptible in any state, realized by executing diagnostic software/firmware routines).

OFF-LINE BIST

1. Deals with testing a system when it is not carrying out its normal functions (Test mode, Non-Real-Time error detection).
2. Testing by using either on-board TPG + Output Response Analyzer (ORA) or Microdiagnostic routines.
3. Structural : Execution based on the structure of the CUT (Explicit fault model – LFSR)
4. Functional : Running based on functional description of CUT (Functional fault model - Diagnostic software).

2.1.2 Test Pattern Generation Techniques

1. Exhaustive : Applying all $2^n$ input combinations, generated by binary counters or complete LFSR.
2. Pseudo exhaustive : Circuit is segmented & each segment is tested exhaustively i.e., Less no. of tests required
3. Logical segmentation : Cone + Sensitized-path

Physical segmentation

Pseudorandom : Not all $2^n$ input combinations, Random patterns generated deterministically & repeatably, pattern with/without replacement, applicable to both combinational and sequential circuits.

Weighted : Non-uniform distribution of 0’s & 1’s, improved fault coverage, using LFSR added with combinational circuits.

Adaptive : Using intermediate results of fault simulation to modify 0’s & 1’s weights, more efficient, more hardware Complexity.

Test Response compression techniques

1. Response compression: A process to form a “signature” from complete output responses.
Signature: Compressed form of saved test results.
Alias: Errorous output when faulty & fault-free sig. are the same.
2. Compression procedure: Composition of test vector applying, results storing and comparison of the faulty & fault free signatures.
3. Compression of Simple hardware implementation and Small performance degradation - No effect on normal circuit behavior (delay, execution time).
High degree of compression - Signature lengths to be a logarithmic factor of responses lengths and Small aliasing errors.
Compression problems will be occurred here, but mainly two things are

1. Existing aliasing errors.
2. Calculating the good circuit signature.

Calculation of good circuit signatures
Golden Unit: Applying the test to good part of the CUT.
Simulation: Simulating the CUT and making sure of having good signature.
Fault Tolerant: Producing copies of CUT and conclude the correct signature by finding the subset which generates the same signature.
One’s count : The no. of times when 1 occurs in each output (counter).
Transition count : The no. of transitions (0 $\Rightarrow$1,1 $\Rightarrow$0) in the output (XOR +counter).
Parity checking: The parity of response string, 0 if even & 1 if odd (XOR + D-FF).
Syndrome checking: the normalized no. of 1’s in output string (k/2**n when k is no. of minterms in an n input circuit), (All possible combination tests).
Signature analysis: Based on redundancy checking (LFSR).

2.2 Factors affecting the choice of BIST

1. Degree of test parallelism
2. Fault coverage
3. Level of packaging
4. Test time
5. Complexity of replaceable unit
6. Factory and field test-and-repair strategy
7. Performance degradation
8. Area overhead

3.1 Accumulator Based 3-Weight Pattern Generation

Generally, the accumulator-based compaction technique uses an accumulator to generate a composite fault signature for a circuit under test. The error coverage for this method has been previously analyzed. We describe an alternative technique for calculating the error coverage of accumulator-based compaction using the asymmetric error model. This technique relies on the central limit theorem of statistics and can be applied to other count-based compaction schemes. The data paths of most contemporary general and special purpose processors include registers, adders and other arithmetic circuits. If these circuits are also used for built-in self-test, the extra area required for embedding testing structures can be cut down efficiently. Several schemes based on accumulators, subtractors, multipliers and shift registers have been proposed and analyzed in the past for parallel test response compaction, whereas some efforts have also been devoted in the bit serial response compaction case.

The utilization of accumulators for time compaction of the responses in built-in self test environments has been studied by various researchers. One of the well-known problems of time compactors is aliasing, i.e. the event that a series of responses containing errors result in a signature equal to that of an error-free response sequence. In this paper we propose a scheme to reduce aliasing in accumulator based compaction environments. With the proposed scheme, the aliasing probability tends to zero, as the number of the patterns of the test set increases.

In general, pseudo random pattern generation requires more patterns than completely deterministic Automatic Test Pattern Generation (ATPG), but obviously, fewer than the exhaustive testing. However, it was found that the stuck-fault coverage rises in a logarithmic fashion towards hundred percentage, but at the cost of enormous numbers of random patterns. On top of it, certain circuits are random pattern resistant circuits in that they do not approach full fault coverage with an unbiased random pattern. Such circuits require extensive insertion of testability hardware or a modification of random pattern generation to ‘weighted pseudo random pattern generation’ in order to obtain an acceptable fault percentage. This desire to achieve higher fault coverage with shorter test lengths and therefore shorter test times led to the invention of the weighted pseudo random pattern generator.

The implementation of the weighted-pattern generation scheme is based on the full adder truth table, presented in Table. From Table Cout=Cin. Therefore, in order to transfer the carry input to the carry output, it is enough to set A[i] = NOT B[i]. The proposed scheme is based on this observation.

![Fig 1: TRUTH TABLE FOR FULL ADDER](image)

3.2 ACCUMULATOR CELL

The main object of the weighted pattern generation is an accumulator cell. To implement the accumulator in the proposed weighted pattern generation scheme is based on presented in Fig
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 the above figure, we assume that the set and reset are active high signals and at the same time the set and reset are used to without loss of generality. And at the time, the respective cell of another register $B[i]$ is also occurred. For this accumulator cell, one out of three configurations can be utilized, as shown in Fig.

In Fig. we present the configuration that drives the CUT inputs. When $A[i]=1$ is required, So the set[i]=1 and reset[i]=0 and hence $A[i]=1$ and $B[i]=0$. Then the output is equal to 1, and $Cin$ is equal to $Cout$. i.e., the $Cin$ is transferred to the $Cout$. And similarly, When $A[i]=0$ is required, So the set[i]=0 and reset[i]=1 and hence $A[i]=0$ and $B[i]=1$. Then the output is equal to 0, and here $Cin$ is equal to $Cout$. i.e., the $Cin$ is transferred to the $Cout$.

When $A[i]=\text{"-"}$ is required, so the 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.

### 3.3 Linear Feedback Shift Registers

The LFSR are the basic building blocks of the pseudo random test pattern generators. In unbiased pseudo random testing, the outputs from the LFSR is fed directly to the CUT and thus the no. of LFSR stages required is equal to the number of inputs to the CUT. For a weighted pseudo random testing we however require much more LFSR stages than the inputs to the CUT. This is so because each weighted bit usually requires more than one equi-probable bit coming in from an LFSR stage for the generation of its weighted bit.

Now we assume that for both the unbiased and the weighted case we have the total number of LFSR shift registers required for each of the CUTs.

### 3.4 Pattern Generation

In the last section we see how, depending upon the no. of inputs of the CUT and the associated probabilities we can make an LFSR configuration for both, the unbiased and the weighted pseudo random testing part. We do this by writing a Verilog file with the entire configuration written in it. After that we run the Verilog simulator and get the raw patterns in a file. This reads the patterns generated from an LFSR working for weighted pseudo random testing.

The general configuration of the proposed scheme is presented. 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.

All schemes require the application of the session counter, required to alter among the different weight sessions. The number of test patterns applied and the proposed scheme is the same, since the test application algorithms that have been invented and applied, Methods to generate weighted pseudo-random patterns for combinational circuits that can be extended to sequential circuits this method is based on the use of three weights, 0, 0.5 and 1. A weight assignment associates one of these weights with every primary input of the circuit. A preselected number of patterns N is applied under every weight assignment. A weight of 0.5 assigned to an input i by a weight assignment w implies that pseudo-random patterns are applied to input i while N test patterns are applied to the circuit; a weight of 0 assigned to input i implies that input i is held at 0 constantly for the N test patterns and a weight of 1 assigned to input i implies that input i is held at 1 constantly for the N test patterns. The weight assignments are based on a deterministic test set. Each weight assignment is obtained by intersecting a subset of deterministic test patterns. The intersection of identical values, 0 or 1, yields a weight of 0 or 1, respectively.

The intersection of different values yields an unspecified value ($x$), which is translated into a weight of 0.5. The intersection of test subsequences yielded weight assignments that were used in a similar way to the ones for combinational circuits. However, for sequential circuits, the intersection of a subset of test subsequences of length M.
results in M weight assignments that have to be used consecutively, and changed at every time unit. The need to change the weight assignment at every time unit is undesirable.

Fig 4: Accumulator cell

Fig 5: D-Flip Flop

Fig 6: Full Adder

Fig 7: Top Module
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-based 3-weight pattern generation technique and it indicates that the hardware overhead of the proposed scheme is lower, while at the same time no redesign of the accumulator is imposed, thus resulting in reduction 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 and reveal that the proposed scheme results in significant decrease in hardware overhead and the amount of time it required to generate the results is very low.

6 REFERENCES
