International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
NCACS-2015 Conference Proceedings

Low Power Concurrent Online Build in Self Testing

[1]H. Sribhuvaneshwari, [2] S. Selvi
[1]Final year M.E (VLSI DESIGN), [2]Assistant Professor, Dept of ECE,
M.Kumarasamy College of Engineering, Karur, Tamil Nadu, India.

Abstract — **Input vector monitoring concurrent (BIST) testing schemes perform testing during the normal operation of the taken circuit without imposing the circuit to be set in offline to perform the test. Testing schemes are evaluated based on the hardware overhead and the concurrent test latency (CTL), i.e., the time required for the test to complete, whereas the circuit operates normally. In this brief, we present a novel input vector monitoring concurrent BIST scheme, which is based on the concept of monitoring a set (called window) of vectors which are reaching the circuit inputs during the normal operation, and the static-RAM like structure is used to store the relative location of the vectors that reaches the circuit input in the examined window; the proposed testing scheme is shown to perform better than earlier testing schemes with in terms of the hardware overhead and CTL tradeoff.**

Index Terms - Built-in self-test, design for testability, testing.

I.INTRODUCTION

Built-in self test (BIST) techniques is a class of schemes that gives the capability of performing at-speed of testing with high fault coverage, whereas parallely they can relax the reliance of expensive external testing equipment. Hence, they constitute an important solution to the problem of VLSI testing devices [1]. BIST is typically classified into offline and online. Offline architectures perform in either mode of operation they are normal mode (during which the BIST circuitry is idle) and test mode. During the test mode, the inputs are generated by the test generator module are given to the inputs of the circuit under test (CUT) and the responses are captured by a response verifier (RV). Therefore, to perform the test, the normal function of the CUT the performance of the system in which the circuit is added. Input vector monitoring concurrent BIST is proposed to avoid performance degradation.This architecture test the CUT concurrently with its normal function by exploiting input vectors reaching the inputs of the CUT; if the incoming vector belongs to the set called active test set, then the RV is enabled to capture the response of CUT. The CUT consists of n inputs and m outputs and they are tested exhaustively; hence, the test set size N is $2n$. The technique can operate in either mode of operation that is normal and test mode, depending on the value incoming vector the signal is labelled as *T/N*. During the normal mode, vector that drives the inputs of the circuit which is to be tested. (denoted by $d[n:1]$) is driven from normal input vector ($A[n:1]$).

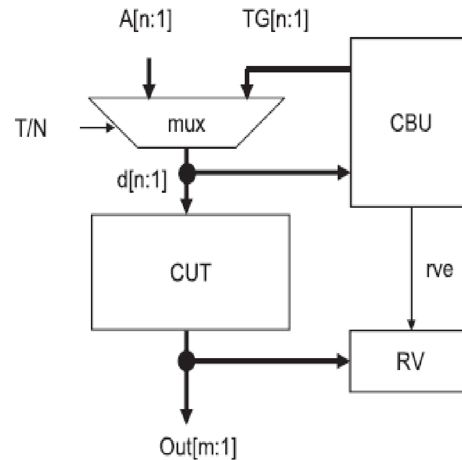

Fig. 1 Input vector monitoring concurrent BIST.

A is driven to the concurrent BIST unit (CBU), where it is compared with active test set. If it is found that A is similar to any one of the vectors in active test set, we say that hit has occurred. In this case, A will be removed from the active test set after that the signal response verifier enable is given to enable the m-stage of the RV to capture the CUT response. When all the input vectors have performed hit, then the contents of RV are examined. During the test mode, the inputs of the CUT are driven from CBU outputs which is denoted as denoted TG[n:1]. The concurrent test latency (CTL) of input vector monitoring is the mean time (counted either in numbers of clock cycles or units of time) required to complete test performance while the circuit performance in normal mode.BIST utilizes a method called *Test Pattern Generator (TPG)* to generate the required test patterns which are applied to the number of inputs of the *Circuit Under Test (CUT).* In off-line BIST, the normal function of the circuit under test is stalled to do the test. Thus, if the idea is of critical importance for the operation of the circuit, the entire circuit performance is degraded. To avoid this common problem performance degradation, input vector monitoring concurrent BIST method have been proposed which will exploit input vectors which are arriving at the inputs of the CUT during the normal operation.
In brief, an input vector monitoring concurrent BIST is proposed, which will compares favourably to the previously proposed schemes in order to the hardware overhead/CTL trade off. We compare the proposed scheme with previous input vector monitoring concurrent BIST schemes. Case study

for concurrent testing of ROM modules are presented. Finally, summarizes the conclusion of this idea.

II. EXISISTING SCHEME

Consider a combinational CUT with n inputs shown in Fig hence possible input vectors for this particular CUT is $2n$. This scheme is based on the concept of monitoring a window of vectors, whose size is denoted as W, with $W = 2w$, here w is an integer $w < n$. Every moment, test vectors that belonging to the window are monitored. If a vector performs hit, then RV is enabled. The bits of input vector are separated into two sets comprising of w and k bits, respectively, where $w + k = n$. The k (which has high order) bits of the input vector will show whether the input vector that belongs to the window under consideration. The w remaining bits will show the relative location of incoming vector in the current window. If any of the incoming vector belongs to current window and has not received during the examination of current window, we can say that the vector has performed hit then the RV is clocked to capture CUT's response to the vector. When all the vectors that belong to current window have reached to the inputs of CUT, we proceed to test the next window. The logic module implementing the idea is shown. It operates in two modes, normal or test mode, depending on the signal value the signal is named as *T/N*. If *T/N* = 0 (normal mode) the inputs of CUT are driven by normal input vector. The inputs of CUT are given to the CBU. The k (high order) bits are given to the inputs of k-stage comparator; other inputs of the k comparator are driven by outputs of a k test generator TG.

1. Pattern coverage:
= (Detected Pattern) / (Total Test Pattern)
2. BIST time (the number of required clock cycles including reset and reseeding)
= Applied test cycles + Seed number * (1 + Total size of LFSRs) = last CYCLE
3. Total LFSR size
= Lm + Lm-1 + … + L1
4. The total data volume of the seeds (in bits)
= Seed number * Total size of LFSRs
5. The upper bound of the number of seeds in use
= 0.2 * Total Test Pattern
6. Performance (Run time, Memory usage)

This scheme uses a modified decoder (denoted by m_dec) and a logic module structure which is based on a static-RAM-like cell.The design of the m_dec module for the window size w = 3 is shown and operates as follows. When the test generator enable (tge) is enabled, all the outputs of the decoder are equals to one. When the comparator (cmp) is disabled (and the tge is not enabled) and all the outputs are disabled. Design of the logic module is shown. When the tge signal is disabled and cmp is enabled, the decoder module operates as a normal function of decoding structure.

The generation of this concurrent BIST test set comprises of four steps:

1) Generation of the partially specified test set with as minimum specified bits as possible and a strict limit on the number of specified bits per pattern.
2) Selection of patterns from this test set such that the targeted fault efficiency is achieved.
3) Selection of a subset of output values to be compared.
4) Generation of the input/output relation of the monitor.

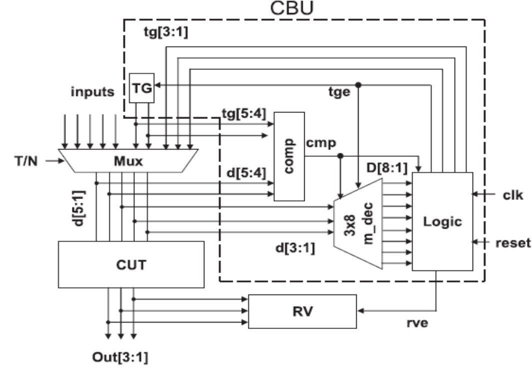

Fig. 2 Circuit with modified decoder and logic module

The logic module comprises of W cells (operating in a fashion which is similar to function of the SRAM cell), w-stage, two D flip-flops, a counter (whose size w = log2W) and a sense amplifier. The overflow signal from the counter drives the tge signal through the unit flip-flop. The signals and the clock (clk) are enabled during both in active low and high of the clock signals, respectively. In this sequel, we assume a clock which is active during the second half cycle of the time period, as shown. In this sequel, we describe the logic module operation presenting the following cases:

1) reset of the logic module
2) hit of the vector (i.e., a vector that belongs in active window and reaches CUT inputs for the very first time)
3) a vector which belongs to the current window reaches the CUT inputs but not first time
4) tge operation (i.e., when all cells of the window are filled then we will proceed to test the next window).

It is not practical to implement a big LFSR to test a circuit with large number of inputs. It is because such implementation often results in lower F.C. and longer BIST time. A good partition of LFSR will improve F.C. and BIST time. Figure 3 shows a CUT with k-bit inputs (Ik, Ik-1, …, I1). The LFSR is partitioned into m smaller LFSRs (LFSR m, LFSRm-1, …, LFSR1). Each LFSR is with a size of Lm, Lm-1, …, L1, respectively, and hence the total size of the LFSRs is Lm+Lm-1+…+ L1 = k.

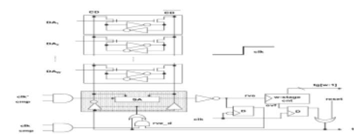

Fig. 3 Design of the logic module.

A. Reset of the Logic Module

At the initial stage of the operation, the module will be reset via the external reset signal. When reset is given, then the tge signal is enabled and decoder outputs are enabled. Hence, DA1,DA2, . . . , DAW are equal to one; furthermore, the CD_ signal also enabled; therefore, one is written in the right side

of the cell and a zero value is written to the left side of the cell.

B. Hit of the Vector (i.e., Vector that Belongs in the Active Window and Reaches the CUT for the First Time)

During the normal mode, inputs to the CUT are driven from normal inputs. The *n* inputs are driven to CBU as follows: the low-order *w* inputs are driven to the decoder inputs; the higher-order *k* inputs are driven to the comparator inputs. When a vector belongs to the current window and reaches the inputs of the CUT, then the comparator is enabled and one of the outputs of the decoder is enabled. During the first half of the clock cycle (clk_ and cmp are enabled) the addressed cell is read; because the read value is zero, the *w*-stage counter is triggered via the NOT gate with output the response verifier enable (rve) signal. During the second half of the operation.

C. Vector That Belongs to the Current Window Reaches CUT Inputs But Not for the Very First Time

If the cell corresponds to the incoming vector have a one (i.e., the respective vector has reached the CUT input during the testing of the current window before), the rve signal is disabled during the first half of the given clock cycle; hence, the *w*-stage counter is not triggered and the AND gate is disabled during the second half of the same clock cycle.

D. tge Operation (i.e., All Cells of the Window are Filled then We Will Proceed to Examine the Next Window)

When all the cells are full (whose value equals to one), then the value of *w*-stage counter is all one. Hence, the activation of the rve signal in the logic module causes the counter to overflow; hence in the next clock cycle (through the unit flop delay) the tge signal is enabled and all the cells (because all the outputs of the decoder of are enabled) are set to null. When switching from the normal mode to test mode, then the *w*-stage counter is reset. During the test mode, the *w*-bit outputs of the counter is given to the CUT inputs. The output of the counter are used to address the cell. If the cell is empty (reset), it will be filled (set) then the RV will be enabled. Otherwise, the cell will remains full and the RV is disabled.

III PROPOSED SCHEME

Saboteurs and mutants

Two different approaches can be used to modify the initial description of the circuit. The first one consists in modifying the structure of the description by adding, between the existing blocks, some additional blocks able to insert some kinds of faults. Such modifications are conceptually quite easy and require only to modify some interconnections in the initial description. However, by consequence, the saboteurs can only inject faults on these interconnections and it is almost impossible to inject higher-level (behavioural) errors or to modify signals within the initial blocks, that is required for example to modify the value of memorised signals or variables. In such cases, some blocks in the initial description have to be directly modified, that is more difficult but much more powerful. In this case, the modified description of the block is called a **mutant**.

Fault Injection Using Mutants

A mutant is a component that replaces another component. While inactive, it works like the original component, but when it is activated, it behaves like the component in presence of faults. The mutation can be made in three ways:

•By adding saboteurs to structural model descriptions.
•By modifying structural descriptions replacing sub-components.
•By modifying syntactical structures of behavioural descriptions.

There can exists lots of possible mutations in a VHDL model, so representative subsets of faults at logical and RT levels must be considered replacing the values of conditions in if and case statements (called stuck-then, stuck-else, dead clause, etc.), disturbing assignment statements (assignment control, global stuck-data, etc.), disturbing operators in expressions (micro-operation, local stuck-data), etc.

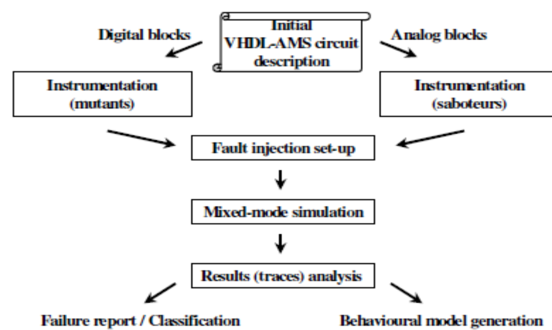

Fig .4 Flow of fault injection

Generation of mutants

The injection of bit-flips in high-level descriptions of digital blocks, as presented, uses such mutants. A *mutation* of a component description can be carried out in several ways by (i) manually mutating it, (ii) inserting saboteurs into it and (iii) manually selecting mutation rules for the automatic mutation. A mutation rule states where the mutation must occur, and what modification to make. The only rules available in MEFISTO are those that map one VHDL grammar rule to another one. A mutation rule can make calls to other mutation rules, in a recursive manner. Indeed, the mutation of an entity containing instances (i.e., components) of other entities may be obtained by the mutation of any of these components. Examples of fault models possible to describe as mutation rules are found in [18] that divides behavioural-level fault models into eight fault classes: Stuck-Then, Stuck-Else, Assignment Control, Dead Process, Dead Clause, Micro operation, Local Stuck-data and Global Stuck-data.

III. CALCULATION OF THE HARDWARE OVERHEAD

The hardware overhead of MHSAT, OISAT $(K = 2k)$, R-CBIST, w-MCBIST, and the SWIM scheme. The cells used

are two-input XOR gate (XOR2), n-input AND gate(ANDn), n-input NAND gate (NANDn), n-input OR gate (Rn), n-input NOR gate (NORn), DFF, FA and two-to-one multiplexer (MUX21). CTL is presented (in time units, i.e., seconds) as a function of the hardware overhead (in gate equivalents) for R-CBIST, w-MCBIST, SWIM, and the proposed architecture. A CUT with n = 16 inputs and m = 16 outputs has been considered..

IV.CASE STUDY: COMPARATIVE CONCURRENT TESTING OF VARIOUS ROM MODULES

ROM modules require high-quality of testing because they constitute critical blocks in many complex circuits, therefore testing schemes for the ROMs use exhaustive application of input patterns, which has been proved to cover all the logically testable combinational faults [14]. For the calculation, we used to considered a ROM cell which is equivalent to ¼ gate (as in [13]). For the case considered in Fig (a 64 k × 16 word memory), the hardware overhead of the ROM is calculated by multiplying the number of cells (64 k × 16 = 65 536 × 16 = 1 048 576) with 1/4, giving 262 144 gates. The percentage of hardware overhead of this proposed scheme as a function of the CTL assuming a 100-MHz clock. We can observe that the concurrent testing can be completed within <4 s with <0.4% overhead. It should be noted that, because of problems in the layout, the actual area overhead may be high. We compare the w-MCBIST, SWIM, and the proposed scheme for the concurrent testing of ROMs with representative sizes. We have not considered R-CBIST in these comparisons, because for these values of the CTL, the R-CBIST scheme does not give favourable results, as shown in Fig. 6. For the calculations, we have considered ROMs with 16-bit words and a 100-MHz clock. In Table III, for every ROM size, we present a group of six rows. In the first row of each group, we present the CTL (s)1) The hardware overhead for the proposed scheme is lower than various other testing schemes.2) The decrease in area that is hardware overhead obtains higher as the CTL decreases; for example, in the 256-k ROM group, the decrease (compared with SWIM) is 11.11% when a CTL = 50.94 s is required, it climbs up to 38.46% when the required CTL is ∼5 s.

RESULT

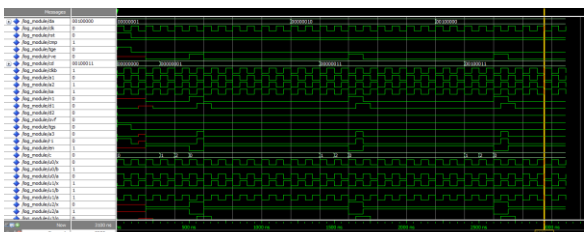

Fig. 8 Result for existing method

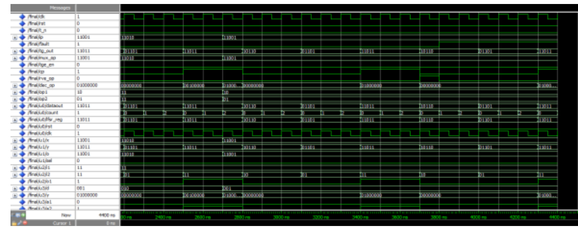

Fig. 9 Result for modified decoder

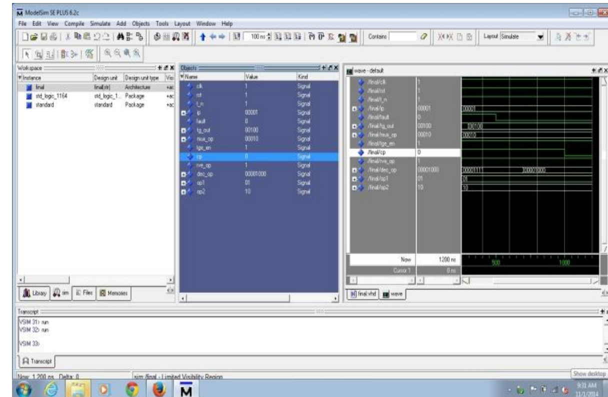

Fig.10 Result for proposed method

VI. CONCLUSION

BIST schemes gives an attractive solution for the problem of testing VLSI devices. Input vector monitoring concurrent BIST schemes perform testing during the circuit normal operation without imposing a need to set the circuit offline to perform the test, therefore they can circumvent problems appearing in offline BIST techniques. The evaluation criteria for the class of testing schemes are the hardware overhead and the CTL, i.e., the time required for the test to complete, when the circuit operates in its normal mode. In this brief, a input vector monitoring concurrent BIST architecture has been presented, based on the usage of a SRAM-cell like structure for storing the information data, whether an input vector has appeared or not during normal operation. The proposed scheme is shown to be more efficient than previously proposed input vector monitoring concurrent BIST techniques with respect to the hardware overhead and CTL. In this project, we mainly introduced new methods to implement and use of saboteurs and mutants into VHDL models. The new models of saboteurs prevent some of the problems that the previously had. These problems prevented the automatic insertion. Moreover, the new models have been implemented in a way that they diminish the hardware overhead, by reducing the number of signals that is required to manage bidirectional saboteurs. Another enhancement is respect to prior models that they allow injecting of more fault models. The advantages of the new proposal is to implement mutants are especially relevant. These saboteurs and mutants have been applied to example circuits at gate level and register level.

REFERENCES

[1] J. Rajski and J. Tyszer, "Test responses compaction in accumulators with rotate carry adders," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 12, no. 4, pp. 531–539, Apr. 1993.

[2] I. Voyiatzis, "On reducing aliasing in accumulator-based compaction," in *Proc. Int. Conf. DTIS*, Mar. 2008, pp. 1–12.

[3] L. R. Huang, J. Y. Jou, and S. Y. Kuo, "Gauss-eliminationbased generation of multiple seed-polynomial pairs for LFSR," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 16, no. 9, pp. 1015–1024, Sep. 1997.

[4] Y. Zorian and A. Ivanov, "An effective BIST scheme for ROM's," *IEEE Trans. Comput.*, vol. 41, no. 5, pp. 646–653, May 1992.

[5] R. Sharma and K. K. Saluja, "Theory, analysis and implementation of an on-line BIST technique," *VLSI Design*, vol. 1, no. 1, pp. 9–22, 1993.

[6] K. K. Saluja, R. Sharma, and C. R. Kime, "Concurrent comparative built-in testing of digital circuits," Dept. Electr. Comput. Eng., Univ. Wisconsin, Madison, WI, USA, Tech. Rep. ECE-8711, 1986.

[7] I. Voyiatzis, T. Haniotakis, C. Efstathiou, and H. Antonopoulou, "A concurrent BIST architecture based on monitoring square windows," in *Proc. 5th Int. Conf. DTIS*, Mar. 2010, pp. 1–6.

[8] M. A. Kochte, C. Zoellin, and H.-J. Wunderlich, "Concurrent self-test with partially specified patterns for low test latency and overhead," in *Proc. 14th Eur. Test Symp.*, May 2009, pp. 53–58.

[9] S. Almukhaizim and Y. Makris, "Concurrent error detection methods for asynchronous burst mode machines," *IEEE Trans. Comput.*, vol. 56, no. 6, pp. 785–798, Jun. 2007.

[10] S. Almukhaizim, P. Drineas, and Y. Makris, "Entropy-driven parity tree selection for low-cost concurrent error detection," *IEEE Trans. Comput.Aided Design Integr. Circuits Syst.*, vol. 25, no. 8, pp. 1547–1554,Aug. 2006.