A NOVEL RESEEDING MECHANISM FOR PSEUDO-RANDOM TESTING OF VLSI CIRCUITS

Jiann-Chyi Rau, Ying-Fu Ho, and Po-Han Wu
Department of Electrical Engineering, Tamkang University
151, Ying-Chuan Rd. Tamsui, Taipei Hsien 251, Taiwan, R.O.C
{jcrau, yfho, phwu}@ee.tku.edu.tw

***ABSTRACT*—DURING BUILT-IN SELF-TEST (BIST), THE SET OF PATTERNS GENERATED BY A PSEUDO-RANDOM PATTERN GENERATOR MAY NOT PROVIDE SUFFICIENTLY HIGH FAULT COVERAGE AND MANY PATTERNS WERE UNDETECTED FAULT (USELESS PATTERNS). IN ORDER TO REDUCE THE TEST TIME, WE CAN REMOVE USELESS PATTERNS OR CHANGE FROM THEM TO USEFUL PATTERNS (FAULT DROPPING). IN THIS PAPER, WE RESEED, MODIFY THE PSEUDO-RANDOM AND USE AN ADDITIONAL BIT COUNTER TO IMPROVE TEST LENGTH AND ACHIEVE HIGH FAULT COVERAGE. THE FACT THAT A RANDOM TEST SET CONTAINS USELESS PATTERNS, SO WE PRESENT A TECHNIQUE, INCLUDING BOTH RESEEDING AND BIT MODIFYING TO REMOVE USELESS PATTERNS OR CHANGE FROM THEM TO USEFUL PATTERNS, AND WHEN THE PATTERNS CHANGE, WE PICK OUT NUMBER OF DIFFERENT LESS BIT, LEADING TO VERY SHORT TEST LENGTH. THE TECHNIQUE WE PRESENT IS APPLICABLE FOR SINGLE-STUCK-AT FAULTS. THE SEEDS WE USE ARE DETERMINISTIC SO 100% FAULTS COVERAGE CAN BE ACHIEVE.**

I. INTRODUCTION

Modern design and package technologies make external testing increasingly difficult and the built-in self-test (BIST) has emerged as a promising solution to the VLSI testing problem. BIST is a design for testability methodology aimed at detecting faulty components in a system by incorporating test logic on-chip. The main components of a BIST scheme are the test pattern generator (TPG), the response compactor, and the signature analyzer. The test generator applies a sequence of patterns to the circuit under test (CUT), the responses are compacted into a signature by the response compactor, and the signature is compared to a fault-free reference value.

Many digital circuits contain random- pattern-resistant (r.p.r) faults that limit the coverage of pseudo-random testing [4]. The r.p.r faults are faults with low detectability (Few patterns detect them). Several techniques have been suggested for enhancing the fault coverage achieved with BIST. These techniques can be classified as:(1) Modifying the circuit under test (CUT) by test point insertion [4][13], or by redesigning the CUT [11][3], to improve the fault detection probabilities. (2)*Weighted pseudo-random patterns*, where the random patterns are biased using extra logic to increase the probability of detecting r.p.r fault [5]. (3) Mixed-mode testing where the circuit is tested in two phases. In the first phase, pseudo-random patterns are applied. In the second phase, deterministic patterns are applied to target the undetected faults [7][14].

This paper use an additional bit counter and modifying circuit in which deterministic test cubes are embedded in the pseudo-random sequence of bits. The modifying circuit is added at the serial output of the LFSR to modify the pseudo-random bit sequence so that the useful patterns will be obtained. This is accomplished by "inversion" certain bits in the sequence. The additional bit counter is used to control CUT. When the bit counter reaches zero, it means that the test pattern is loaded into the scan chains. A procedure is described for designing the modifying-bit sequence generator and using an additional bit counter in a way that decrease test length and area overhead with obtain high fault coverage. Our approach was due to addition modifying circuit and additional bit counter. It guarantees that certain test cube will be applied to the circuit-under-test during a specified test length. Figure.1 shows the global operations.

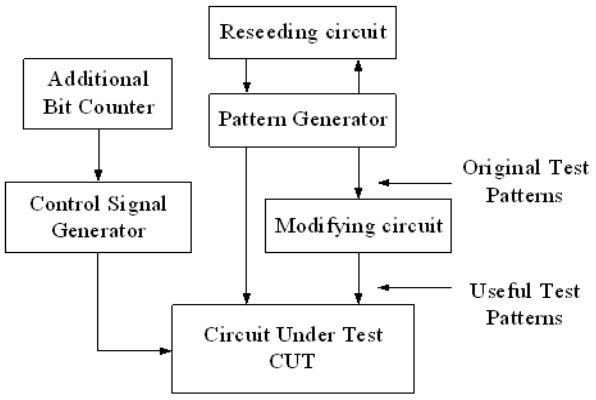

Figure 1: Block Diagram for Generation of useful patterns

In the proposed scheme of this paper, we just use a little storage for additional bit counter, and that use fewer number of test length and seed can achieve desire high fault coverage. The paper is organized as follows. Section 2

introduces the related literature. Section 3 describes the modifying-bit architecture, additional bit counter and the procedure for obtaining useful patterns which use fewer number of test length. Section 4 shows the simulation results and Section 5 concludes the paper.

II. RELATED WORK

In serial BIST, deterministic patterns are applied after a random testing to reduce number of the pattern. The deterministic pattern are loaded into the LFSR and then expended into the desire patterns in the scan chain.

The [10] presented a reseeding-based technique that improves the encoding efficiency by using variable-length seeds together with a multiple polynomial LFSR. The technique reuses part of the scan chain flip-flop in expanding the seeds.

In [12], their technique, random patterns that don't detect r.p.r faults are mapped to ATPG generated cubes through combinational logic. The mapping is performed in two phases, the pseudo-random patterns are identified in the first step, and the ATPG cubes are loaded in the second step. Several iterative minimization heuristics are applied to reduce the area overhead of the mapping logic.

In [1], they loaded new seed by putting the LFSR in the state that precedes the seed value, so that at the next clock pulse, the new seed is in the LFSR, and their technique is based on deterministic seeds which expand into ATPG patterns so 100% fault coverage can be achieved. The algorithm they present is based on the following strategies: (1) generate ATPG patterns for faults that were not detected with pseudo-random patterns and calculate seeds for these patterns, (2) when a seed is loaded into the LFSR, let the LFSR run in autonomous mode for sometime because there is a chance that some of the ATPG patterns will drop more faults so that some of the ATPG patterns are not needed, (3) as long as pseudo-random patterns don't detect faults, the LFSR should be loaded with a new pattern.

The above schemes use seeds that don't particularly target undetected faults, so the test length would be increased. Our technique is based on deterministic seeds which expand into ATPG patterns so high fault coverage can be achieved and reduced the test length.

A. Built-In Reseeding

In [1], Reseeding refers to loading the LFSR with a seed that expands into a pre computed test pattern. The operation of the reseeding circuit is as follows: the LFSR starts running in autonomous mode for sometime according to the reseeding algorithm. Once it is time for reseeding, a seed is loaded into the LFSR, which then goes back to the autonomous mode and so on and so forth until the desired coverage is achieved. The new seed is in the LFSR. Their technique use muxed flip-flops as shown in Figure 2(b). By activating the select line of a given mux, the logic value in the corresponding LFSR stage is inverted.

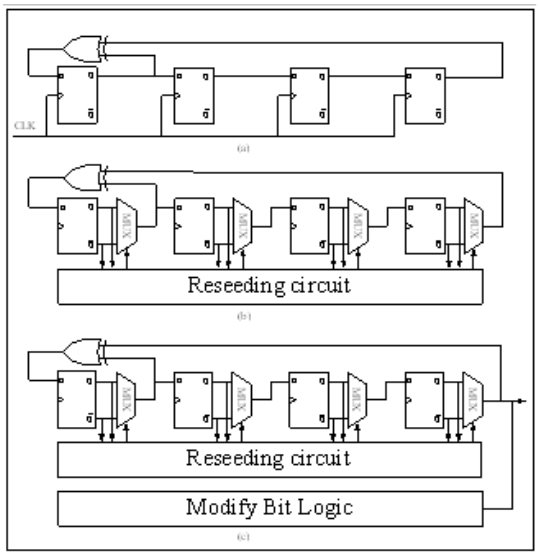

Figure 2: (a) A standard 4-stage LFSR
(b) 4-stage LFSR with reseeding circuit
(c) 4-stage reseeding with modifying circuit

III. OUR EMBEDDING ALGORITHM

A. Additional Bit counter

In the common logic BIST architecture, if we want to disable the Scan Enable signal for capturing, we can use a bit counter. Generally speaking, the bit counter loaded with the value that corresponds to the length of the scan chain for every pattern. The bit counter is decremented by 1 at each clock cycle. When the bit counter counts to zero, it means that the test pattern is loaded into the scan chains, so Scan Enable signal is disabled for one clock cycle, and we can capture.

In the PRPG, if we want to reach the desired seed. We need to load the bit counter register with different values corresponding to the number of cycles before the next capture. The value corresponds to the length of the scan chains plus the distance of the two useful patterns in the LFSR sequence.

As an example in Figure 3, assume that the patterns 1000 and 1110 are useful and the first useful pattern is in the FLSR, so we load the bit counter register with 6. After 4 clock cycles, the first pattern are loaded into the scan chain, and at clock cycle 6(the length of the scan chains plus the distance of the two useful patterns), the second pattern are loaded into scan chain as shown in Figure 3(a). For another example as shown in Figure 3(b), assume that the ATPG tools generates the following two patterns: (X1XX1X1XX0, X10XXXX1XX). We can generate two seeds (1000, 1110) for the two patterns [2], so we load the bit counter register with 12. Assume the starting state of LFSR is 1000. After 10 clock cycles, the first pattern are loaded into the scan chain, and at clock cycle 12, the second pattern are loaded into scan, so the test length can be reduced (as long as the distance of the two useful patterns less than the length of the scan chain).

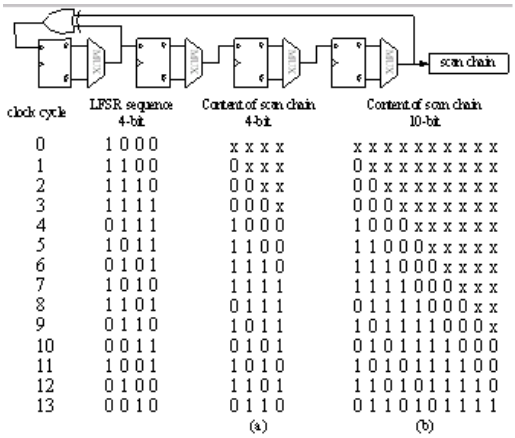

Figure 3 : Example of additional bit counter

B. Modifying-Bit Architecture

The ATPG generate test cubes are embedded in a way that guarantees that faults are detected after the test cubes are embedded. This is performed by modified patterns that don't drop any faults. As long as the patterns can be reduce faults, then we would not modify pattern. High fault coverage can obtain by change some bit of undetected patterns.

In the architecture, a bit counter used to choose when to change the bit value of the useless patterns (the different bits values the correlation between useless pattern and test cub). Our technique is based on modify some bits on useless patterns of pseudorandom mode so can shortens the test length; and further, it is decrease number of seeds. The Modify Bit Logic is shown in Figure 4. We use n-input AND gates where n is equal to the number of bits in the bit counter, and the max number of bit counter depend on length of the test cube. We pay the price in hardware overhead. If the total amount of modify bits is k, we just need k Modify Bit Logic.

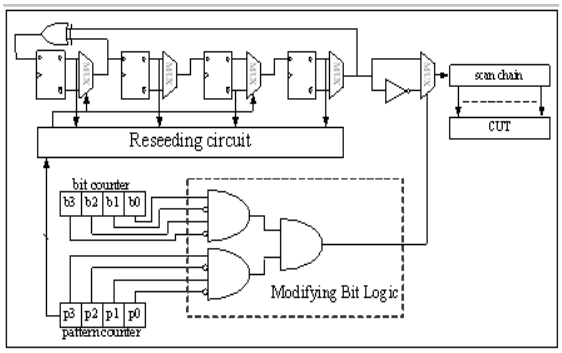

Figure 4: Example modifying with reseeding circuit

In order to reduce the Modify-bit Logic, we must choose a set of useless patterns by comparing number of inverse bits with test cubes, and that pick out the pattern of the less number of different bit from all test cubes by using C language, so that choose a set of patterns that don't decrease any fault (useless patterns) in the circuit-under-test is the most significant.

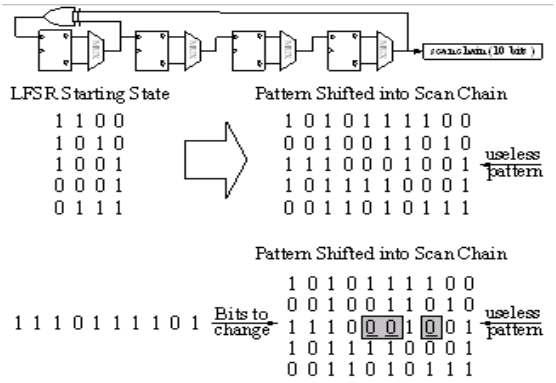

Figure 5 : Obtaining the Test Cubes

The example of the Figure.5 obtaining test cubes shows as follows, the 3rd, 5th and 6th bit of useless pattern must be change into "1" and so can generation ATPG test cubes when the condition established the pattern counter and the bit counter both. The other bits positions of the useless pattern are the same as ATPG test cubes and so only inversion position of the 3rd, 5th and 6th bit of the pattern at pseudorandom sequence of bits that is shifted into the scan chain in order to embed deterministic test cube in the sequence.

C. The Proposed Embedding Algorithm

In this paper, we present is based on the following steps:

Step 1: ATPG tool is used to generate the test cubes and find the position (clock cycle) of the test cube run in pseudorandom mode.

Step 2: We must considering the waste of cycle (position of useless patterns) between the useful patterns at run pseudorandom mode. That is because position of useless patterns will be overwritten by test cube (change some bits). As long as pseudorandom patterns don't drop faults, the seed loaded into the LFSR that could be skipping some useful patterns, and therefore must be regenerate those patterns.

Step 3: If the distance of the two useful patterns is less than the length of the scan chain, we can use an additional bit counter to reduce the test length.

Step 4: The useless patterns of LFSR run in autonomous mode as compare with ATPG test cubes, and count number of different bits position and pick out number of different less bit from all test cubes by using C language in order to minimize hardware overhead.

Step 5: In Step 4, we can find the pattern that need to be changed, so we start to modify the position of the different bits at pseudorandom sequence of bits that is shifted into the scan chain.

Step 6: Lastly, if all ATPG test cubes whole appeared on pseudorandom mode, it means that all test cubes are embedded. Otherwise, loops back to step 4.

Figure 6 shows where the reseeding, modifying circuit and an additional bit counter fit in a system level view of a circuit with an LBIST controller. The pattern counter is used to count the patterns, bit counter is used to count the bits of

pattern applied to the circuit under test and the additional bit counter is used to be loaded with different values corresponding to the number of cycles before the next capture. Complete details about the architecture for built-in reseeding are given in [1].

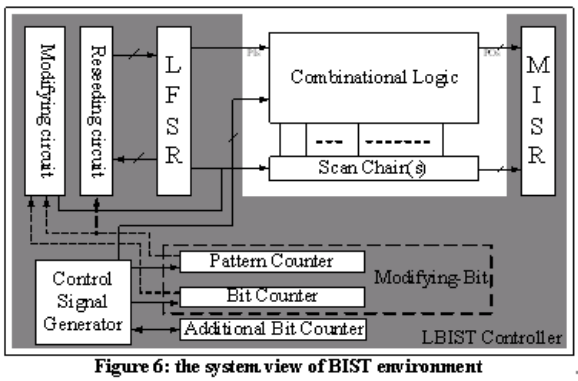

Figure 6: the system view of BIST environment

In a BIST environment, where the LFSR is known in advance and the initial seed and test length by ATPG tool generation test cubes, the reseeding and modifying circuit may take its inputs form the pattern counter instead of the LFSR contents. The modifying circuit requires counting for certain bits of change in the sequence so modifying circuit may take its inputs form the bit counter.

IV. SIMULATION RESULT

In this section we present the results of some simulation experiments. We performed our experiments on some of the ISCAS 89 benchmarks. The characteristics of the benchmarks we used are shown in Table 1. The table shows the number of primary input, number of faults, total amount of modify bits and flip-flop in the scan chain. The BC column lists the sizes of the bit counters and the ABC column lists the sizes of the additional bit counter. The number of changing bits determines the area of the modifying circuit. In our technique, deterministic test cubes were generated using the ATPG tool of the SIS.

circuit	#inps	FFs	#faults	BC	ABC	mount of modify bits
s420_89	19	16	840	6	7	130
s641	35	19	1274	6	7	652
s838_89	35	32	1676	7	8	405
s1494	8	6	2972	5	6	92
s9234	36	211	17350	9	10	5924

Table 1: ISCAS 89 Circuits Used in the Experiments

A. Comparison with previous work

We performed some simulation experiments to compare our technique with [6]. The reseeding with modifying-bit generators and additional bit counter were designed to provide 100% fault coverage of all detectable single stuck-at faults for a test length of fewer patterns.

Table 2 shows the test length, fault coverage and number of seed when our technique is used. The number of seed decrease ranged from 26.7% to 57.1%.

circuit	Test Length		Use number of seed			FC in %	
	[6] calculating efficient seeds	Our	[6]	Our	% Reduction	[6]	Our
S420_89	1000	852	52	32	38.5	94.0	100
	10000					96.4	
s641	1000	1656	15	11	26.7	98.4	100
	10000					99.5	
S838_89	1000	3412	95	65	31.6	85.4	100
	10000					86.2	
s1494	1000	712	14	6	57.1	99.1	100
	10000					100	
s9234	1000	16256	598	415	30.6	84.9	100
	10000					92.8	

Table 2: Comparison of our technique and [6]

V. CONCLUSION

We presented a scheme include built-in modifying-bit and additional bit counter. 100% fault coverage (single-stuck-at fault) can be achieved with our technique without any external testing. We pay the price in hardware overhead in order to decrease test length. Our scheme allows the designer to trade off between the number of seeds and the amount of modifying bit logic.

REFERENCES

[1] Al-Alyamani A., and E. J. McCluskey, "Built-In Reseeding for Serial BIST," *VLSI Test Sympsium*, Apr., 2003.

[2] Al-Alyamani A., S. Mitra, and E.J. McCluskey, "BIST Reseeding with Very Few Seeds," *VLSI Test Sympsium*, Apr., 2003.

[3] Chiang, C.-H., and S.K. Gupta, "Random Pattern Testable Logic Synthesis," *Proc. of International Conference on Computer-Aided Design (ICCAD)*, pp. 125-128, 1994.

[4] Eichelberger, E. B., and E. Lindbloom, "Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test," *IBM Journal of Research and Development*, Vol. 27, No. 3, pp. 265-272, May. 1983.

[5] Eichelberger, E. B., and E. Lindbloom, F. Motica, and J. Waicukauski, "Weighted Random Pattern Testing Apparatus and Method," US Patent 4,801,870, Jan. 89.

[6] Fagot, C., O. Gascuel, P. Girard and C. Landrault, "On Calculating Efficient LFSR Seeds for Built-In Self Test," *Proc. of European Test Workshop*, pp. 7-14, 1999.

[7] Hellebrand, S., B. Reeb, S. Tarnick, and H.-J.Wunderlich, "Pattern Generation for a Deterministic BIST Scheme," *Proc. of International Conference on Computer-Aided Design (ICCAD), pp.88-94, 1995*.

[8] M. Lempel, S.K. Gupta and M.A. Breuer, "*Test Embedding with Discrete Logarithms*," IEEE VLSI Test Symp., pp. 74-78, 1994.

[9] McCluskey, E.J., "Built-In Self-Test Techniques," *IEEE Des. & Test of Comp.*, pp. 21-28, Apr. 85.

[10] Rajski, J., J. Tyszer and N. Zacharia, "Test Data Decompression for Multiple Scan Designs with Boundary Scan," *IEEE Transactions on Computers*, Vol. 47, No. 11, pp.1188-1200, Nov. 1998.

[11] Touba, N.A., and E.J. McCluskey, "Automated Logic Synthesis of Random Pattern Testable Circuits," *Proc. of International Test Conference*, pp. 174-183, 1994.

[12] Touba, N.A., and E.J. McCluskey, "Synthesis of Mapping Logic for Generating Transformed Pseudo-Random Patterns for BIST," *Proc. of International Test Conference*, pp. 674-682, 1995.

[13] Touba, N.A., and E.J. McCluskey, "Test Point Insertion Based on Path Tracing," *Proc. of VLSI Test Symposium*, pp. 2-8, 1996.

[14] Touba, N. and E.J. McCluskey, "Altering Bit Sequence to Contain Predetermined Patterns," US Patent 6,061,818, May, 2000.