IMPLEMENTATION OF HIGH SPEED ACCUMULATOR BASED 3-WEIGHTED PATTERN GENERATION

[1]GUMMA HARI BABU, [2]Mrs.Y.SYAMALA
[1]M.Tech, ECE Dept, E-mail:gumma.babu@gmail.com
[2]Associate Professor, E-mail:coolsyamu@gmail.com

ABSTRACT-*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. In this thesis 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 BIST pattern generation, as well. Comparisons with previously presented schemes indicate that the proposed scheme compares favorably with respect to the required hardware.*

Keywords: *BIST, Accumulator, ATPG, Xilinx, vhdl*

1. INTRODUCTION

Pseudorandom built-in self test (BIST) generators have been widely utilized to test integrated circuits and systems. The arsenal of pseudo-random generators includes, among others, linear feedback shift registers (LFSRs), cellular automata, and accumulators driven by a constant value. For circuits with hard-to-detect faults, a large number of random patterns have to be generated before high fault coverage is achieved. Therefore, weighted pseudorandom techniques have been proposed where inputs are biased by changing the probability of a “0” or a “1” on a given input from 0.5 (for pure pseudorandom tests) to some other value. Weighted random pattern generation methods relying on a single weight assignment usually fail to achieve complete fault coverage using a reasonable number of test patterns since, although the weights are computed to be suitable for most faults, some faults may require long test sequences to be detected with these weight assignments if they do not match their activation and propagation requirements. Multiple weight assignments have been suggested for the case that different faults require different biases of the input combinations applied to the circuit, to ensure that a relatively small number of patterns can detect all faults. Approaches to derive weight assignments for given deterministic tests are attractive since they have the potential to allow complete coverage with a significantly smaller number of test patterns.

In order to minimize the hardware implementation cost, other schemes based on multiple weight assignments utilized weights 0, 1, and 0.5. This approach boils down to keeping some outputs of the generator steady (to either 0 or 1) and letting the remaining outputs change values (pseudo-) randomly (weight 0.5). This approach, apart from reducing the hardware overhead has beneficial effect on the consumed power, since some of the circuit under test (CUT) inputs (those having weight 0 or 1) remain steady during the specific test session. Pomeranz and Reddy proposed a 3-weight pattern generation scheme relying on weights 0, 1, and 0.5. The choice of weights 0, 1, and 0.5 was done in order to minimize the hardware implementation cost. Wang proposed a 3-weight random pattern generator based on scan chains utilizing weights 0, 1, and 0.5, in a way similar. Recently, Zhanget al. renovated the interest in the 3-weight pattern generation schemes, proposing an efficient compaction scheme for the 3-weight patterns 0, 1, and 0.5. From the above we can conclude that 3-weight pattern generation based on weights 0, 1, and 0.5 has practical interest since it combines low implementation cost with low test time. Current VLSI circuits, e.g., data path architectures, or digital signal processing chips commonly contain arithmetic modules [accumulators or arithmetic logic units (ALUs)]. This has fired the idea of 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 over-head and low impact on the circuit normal operating speed. In, Manichet al.presented an accumulator-based test pattern generation scheme that compares favorably to previously proposed schemes. In, it was proved that the test vectors generated by an accumulator whose inputs are 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 the mission logic or by storing additional deterministic test patterns.

2. ACCUMULATOR-BASED3-WEIGHT PATTERN GENERATION

We shall illustrate the idea of an accumulator-based 3-weight pattern generation by means of an example. Let us consider the test set for the c17 ISCAS benchmark given in Table I. Starting from this deterministic test set, in order to apply the 3-weight pattern generation scheme, one of the schemes proposed 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: The weight assignments for these subsets is denotes a weight assignment of 0.5, a “1” indicates that the input is constantly driven by the logic “1” value, and “0” indicates that the input is driven by the logic “0” value. 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 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.

3. DESIGN METHODOLOGY

The implementation of the weighted-pattern generation scheme is based on the full adder truth table, presented in Table II. From Table II 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.

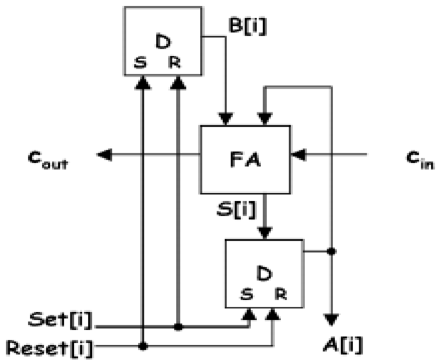


Fig. 3.1. Accumulator cell.

The implementation of the proposed weighted pattern generation scheme is based on the accumulator cell presented in Fig. 3.1, 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 Fig. 3.1, 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 Fig. 3.2.

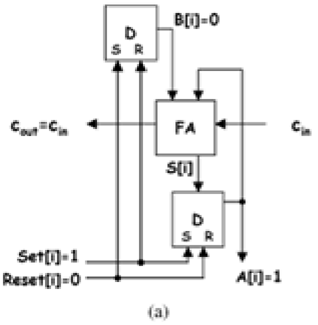
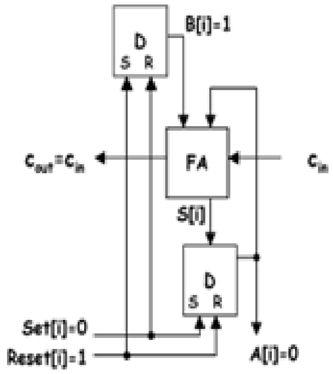


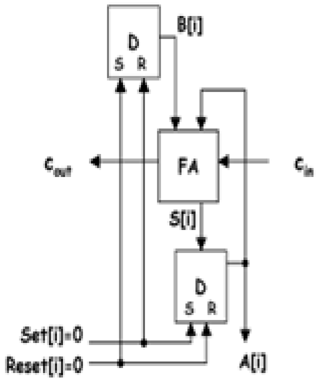
Fig). 3.2(a) Configurations of the accumulator cell of fig 2.1

In Fig. 3.2(a) we present the configuration that drives the CUT inputs when is required. Then the output is equal to 1.



(b)

In Fig. 3.2(b), we present the configuration that drives the CUT inputs. Then, the output is equal to 0 is transferred to it.



(c)

Fig. 3.2(c), we present the configuration that drives the CUT

In Fig. 3.2(c), we present the configuration that drives the CUT. 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.

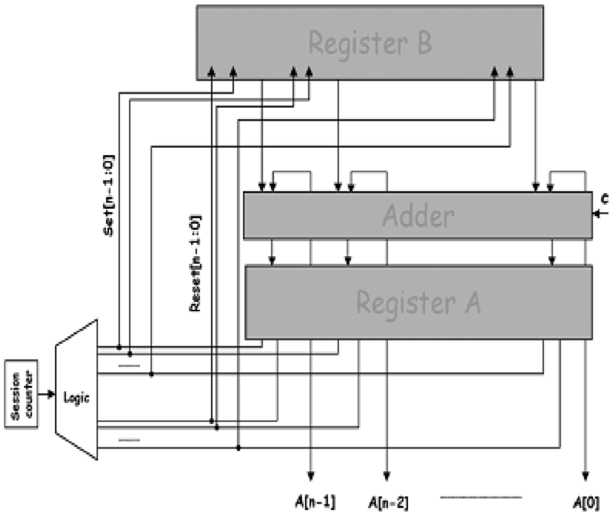


Fig). 3.3 Proposed Scheme

In Fig. 3.3, the general configuration of the proposed scheme is presented. The Logic module provides the Set and Reset signals that drive the S and R inputs of the Register A and Register B in-puts. 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.

4. COMPARISONS

In this section, we shall perform comparisons in three ways. For the proposed scheme, no modification is imposed on the adder of the accumulator. Therefore, there is no impact on the data path timing characteristics.

ATPG	DELAY
LFSR	4.394 NS
EXISTED ACC	7.454ns
PRAPOSED ACC	4.283 NS

Fig 4.1: Delay comparisons of exited and proposed systems

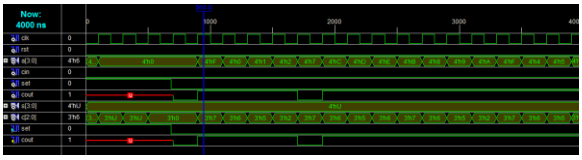
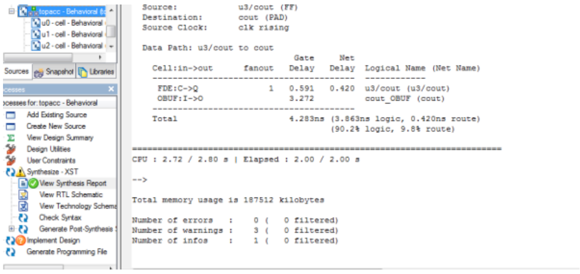


Fig 4.2 : simulation results of proposed ATPG



```
 Source:            u3/cout (FF)
 Destination:       cout (PAD)
 Source Clock:      clk rising

 Data Path: u3/cout to cout
                                Gate     Net
   Cell:in->out      fanout    Delay   Delay  Logical Name (Net Name)
   ----------------------------------------  ------------
    FDE:C->Q              1    0.591   0.420  u3/cout (u3/cout)
    OBUF:I->O                  3.272          cout_OBUF (cout)
   ----------------------------------------
   Total                       4.283ns (3.863ns logic, 0.420ns route)
                                       (90.2% logic, 9.8% route)

=========================================================================
CPU : 2.72 / 2.80 s | Elapsed : 2.00 / 2.00 s

-->

Total memory usage is 187512 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :    3 (   0 filtered)
Number of infos    :    1 (   0 filtered)
```

Fig 4.3: delay report of proposed ATPG

5. 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-based 3-weight pattern generation technique indicate that the proposed scheme is low delay time..All the blocks of proposed system is functionally verified and synthesized by using Spartan 3e Fpga.

6.REFERENCES

[1] P. Bardell, W. McAnney, and J. Savir, Built-In Test For VLSI: Pseudo-random Techniques. New York: Wiley, 1987.

[2] P. Hortensius, R. McLeod, W. Pries, M. Miller, and H. Card, “Cel-lular automata-based pseudorandom generators for built-in self test,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 8, no. 8, pp. 842–859, Aug. 1989.

[3] A. Stroele, “A self test approach using accumulators as test pattern generators,” inProc. Int. Symp. Circuits Syst., 1995, pp. 2120–2123.

[4] H. J. Wunderlich, “Multiple distributions for biased random test pat-terns,” inProc. IEEE Int. Test Conf., 1988, pp. 236–244.

[5] I. Pomeranz and S. M. Reddy, “3 weight pseudo-random test genera-tion 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, Jul. 1993.

[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. 1358–1369, Nov. 1997.

[7] J. Rajski and J. Tyszer, Arithmetic Built-In Self Test For Embedded Systems. Upper Saddle River, NJ: Prentice Hall PTR, 1998.

[8] S. Wang, “Low hardware overhead scan based 3-weight weighted random BIST,” inProc. IEEE Int. Test Conf., 2001, pp. 868–877.

[9] S. Zhang, S. C. Seth, and B. B. Bhattacharya, “Efficient test compaction for pseudo-random testing,” inProc. 14th Asian Test Symp., 2005, pp.337–342