Journal of Scientific & Industrial Research
Vol. 76, March 2017, pp. 149-153

Built-in Self-Test Methodology for System-on-a-Chip Testing

S Sivanantham* and T Tresa

School of Electronics Engineering, VIT University, Vellore – 63201, Tamil Nadu, India

Received 14 December 2015; revised 19 October 2016; accepted 01 January 2017

Built-in Self-Test is a circuit embedded within the design to detect the faults in the System-on-a-Chip circuits. It shrinks the test application time and reduces the cost of external testing equipment. This paper presents a test pattern generation methodology for detection of transition faults using in-circuit arithmetic circuits. Arithmetic circuits are used for pattern generation which utilizes the accumulators in the design itself with the inclusion of additional control unit for pattern generation. This control unit controls the pattern generation circuit to reduce the transition power during testing and ensure better fault coverage. The main focus of this paper is to reduce the area overhead using arithmetic circuits and also to reduce the power of the test pattern generator. The proposed method can be made use in circuits that contains gray code converters. Experimental results show that the proposed technique has lower power consumption and lesser hardware compared to linear generators.

Keywords: Integrated Circuits Testing, System-on-a-Chip, Design for Testability, BIST, Test Pattern Generation, Transition Fault, Arithmetic Circuits

Introduction

Transition fault is a type of fault in which the actual response will not be obtained within the stipulated amount of time. Two-pattern test requires a pair of patterns for detection of faults in testing of System-on-a-Chip (SoC) circuits. This pair comprises of launch vector and capture vector. Launch vector sets the node to the initial value, while the capture vector launches the initial value and propagates the effect to the output node. Several methods for two-pattern generation have been proposed in the literature. Algorithms for pattern generation based includes arithmetic circuits, Linear feedback shift register (LFSRs), Cellular Automata (CA). LFSRs are a class of linear sequential machines built from D flip-flops and modulo-2 adders (XOR gates). A pseudo exhaustive generator can be used to detect transition fault and delay faults in order to get 100% fault coverage with lower hardware overhead[1]. Nan Li *etal*[2] proposed a methodology which uses register with non-linear update (RNLU) to generate the deterministic test patterns. RNLU is based on non-linear LFSR and it is mainly used for test set with high don't care bits. Selecting the best RNLUs to get complete fault coverage is challenging task. An adaptive type technique called a low-power random pattern generator[3] with transition controller which is capable of generating highly correlated test patterns is presented. Adaptive test clock scheme[4] can be used to reduce the test time of an external test applied from automatic test equipment in order to speed up the low activity cycles by keeping the total test power under control. Many techniques to reduce test power with test data volume have been proposed for scan based testing[5,6]. Also, many algorithms have been proposed with LFSR structure to get desired fault coverage. The hardware overhead of LFSR based pattern generation scheme is to be more[7]. Accumulator based pattern generation scheme reduce the area overhead and is quite simple compared to other pattern generation circuits[8, 9, 10]. This paper presents an efficient test pattern generation methodology to utilize the on-chip accumulator circuit for transition fault detection of BIST based SoC circuits.

Architecture

Accumulators with one's compliment adder can be used for generation of pseudo-exhaustive two-pattern tests. This can be utilized for detection of delay and stuck-open faults. Test-pattern generation using the adders available in data-path architecture used in digital signal processing circuit and general purpose processors can be done with no performance degradation. Figure 1 shows a basic configuration of pattern generation using arithmetic circuits. It consists only of accumulators and registers. The input is added to the content of register by the accumulator. The input should be a non-zero

*Author for Correspondence
E-mail: ssivanantham@vit.ac.in

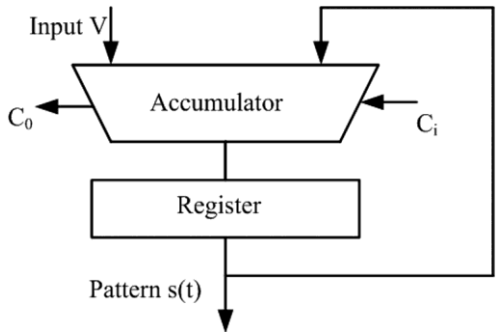


Fig. 1—Accumulator based pattern generation

value to get different patterns; otherwise it may result in a single pattern. Pattern generation using different types of accumulators such as binary adder, ones complement adder and stored carry bit adder was done. Implementation of carry-free accumulators is much easier and can operate at a higher speed than corresponding one's compliment adder. Weighted pseudo random test pattern generation is frequently used in BIST which will reduce the test data volume to get very high fault coverage.

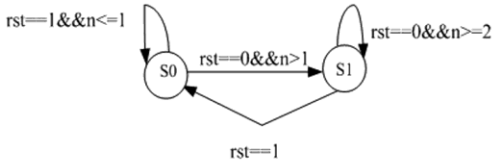
An accumulator based weighted pseudo random pattern generator is proposed to get complete fault coverage for two pattern testing. Weighted accumulator is used to generate the 3 different weights (0, 0.5 and 1) for test patterns. It has low impact on test application time and area overhead. Accumulator based two-pattern testing (ABT) algorithm is implemented by using accumulators whose input are driven by a binary counter or LFSR. The designs which consist of counters and arithmetic circuits, arithmetic pattern generation are used and in other case LFSR is used. Use of accumulators has the advantage that an accumulator does not require to be reconfigured to work in test mode. This algorithm generates an exhaustive *n*-bit pattern with $2^n \times (2^n - 1) + 1$ cycle. The hardware implementation of this algorithm consists of an *n*-stage accumulator, *n*-stage counter and a control module.

Two-pattern generator

The proposed two-pattern generation method uses a circuit that contains gray code converters. Gray code can be referred to as single-distance code that is transition between successive vectors is unity. Multi bit count information can be passed between synchronous systems operating at different clock frequencies by using gray code. One main application of gray code is in implementation of FIFO with different read and writes clock frequencies. A sequence given as $0 \ldots 2^{N-1}$ can be represented in a binary sequence of length *N*, in a pattern that the adjacent integers can be represented in Gray code that differ in only one bit. This property called as adjacency property which is used in our design for generation of patterns with less number of transitions. A double sequence (*N, m*) can be generated for transition fault detection, where *N* is the length of the sequence and *m* is the seed value. The hardware implementation for a two-pattern generation circuit with less transition power will require complicated circuitry. But incorporating Gray code converter with the pattern generation circuit will effectively reduce the transition power. An algorithm proposed in this paper reduces the transition power by reducing number of transitions while also maintaining reduced hardware overhead for test pattern generation. The resulting algorithm is based on the following reasoning. We use a counter and a gray code converter, for which the outputs are accumulated. The gray code converter goes through the sequence from $1 \ldots 2^N$. Each time the accumulator output value reaches the value from which it started, the internal register is incremented and the value is added to the gray code. Here, no feedback circuit is required for pattern generation. The algorithm for this is given in Figure 2 (a) in a *C* like notation. The inputs of accumulator are driven from the outputs of gray converter and *n_gen* circuit. The control module controls for the efficient generation of patterns. The control unit can be represented with two state finite state machines (FSM) as shown in Figure 2 (b). The state of the FSM determines the mode of operation of the accumulator. The first state *S0*, the initial state of pattern generation is the state in which reset is applied and also in which the register *n* is less than one. The accumulator is disabled during this state and the gray value is directly driven as the output of pattern generator. The register *n* increments when the Seed value or the initial value is repeated. The state machine is triggered to next stage *S1* when the register *n* exceeds one. For each repetition of the seed value the register is incremented. At this stage of finite state machine accumulation is enabled. The accumulation of values from output of gray converter and *n_gen* takes place. Accumulator becomes functional in this state of operation. This minimizes the probability of repetition of patterns and to get all possible combinations of two-patterns. Table 1 shows the pattern generated by using the Gray code

```
void ()
int n==0,p,a,seed;
While (1) {            //Check for seed value repetition
if (a==seed) {                 //and increments register n
n++;
}
}                      //if n less than or equal to 1 the
If (n≤1) {             //accumulator is disabled and the
a==p;                          // gray value is given as   output
}
Else {                 //gray n greater than  1,
a=p+n ;}               //accumulator is enabled
```

(a)



(b)

Fig. 2—(a) Algorithm for minimizing transition power and reducing area overhead (b) State diagram representation of control module

Table 1—Pattern for N=16

No.	N	A	No.	N	A
1	0	0	26	2	13
2	0	2	27	2	12
3	0	6	28	2	11
4	0	7	29	2	4
5	0	5	30	2	2
6	0	4	31	2	3
7	0	12	32	2	6
8	0	13	33	3	5
9	0	15	34	3	9
10	0	14	35	3	10
11	0	10	36	3	8
12	0	11	37	3	7
13	0	9	38	3	15
14	0	8	39	3	0
15	0	0	40	3	1
16	0	1	41	3	2
17	0	3	42	3	13
18	1	8	43	3	14
19	2	9	44	3	12
20	2	7	45	3	11
21	2	6	46	3	3
22	2	14	47	3	5
23	2	15	48	4	7
24	2	1	49	4	10
25	2	0	50	4	11

converter. Figure 3 is used to generate *(N, m)* patterns. The value of *n* is incremented for every repetition of the seed value. For value of *n<=1*, the accumulator will be disabled and the gray value is directly given as the output vector. This actually acts as the control signal for the accumulator. This can be signified by a

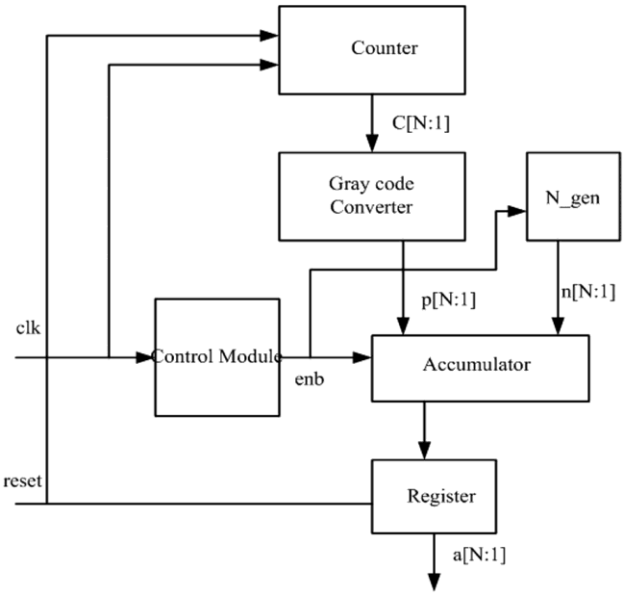


Fig.3—Two-pattern Generator

two state finite state machine, those cycles through state 0 and 1. The state transition occurs from 0 to 1 once the value of *n* exceeds 1. The output of finite state machine (FSM) provides the control signal for the accumulator. Each time *n>=2*, the accumulator is activated. The test pattern generation circuit consists of the control module, accumulator, counter and gray converter. The enable signal (*enb)* is a control signal for both accumulator and gray code converter and it is enabled for $2 \leq n \leq N$ otherwise *enb* is disabled. The module does not require any feedback for accumulator unlike the other methods proposed earlier. The design uses a carry free adder for accumulation as it is faster and efficient.

Hardware evaluation

Hardware implementations of the proposed work with the different methodology presented in the literature that are generating two-patterns are considered in this section in order to evaluate the effectiveness of the proposed work. Multiple input shift register can generate all transitions if 2^n input combinations are available. Generation of 2^n input combinations requires *n*FFs and *n* XOR gate. Thus the hardware overhead of this could be given as

$$HO(n) = n \times DFF + n\ XOR \quad \ldots (1)$$

LFSR or Cellular Automata can be used to generate two-patterns. This technique requires an LFSR or CA with *2n* stages for an *n*-input CUT. It is assumed to

have *n*-inputs at input of an *n*-input CUT, so the hardware overhead of the design could be given as

$$HO(n) = n \times DFF \quad \ldots (2)$$

Carry-free accumulators with inputs from either counter or LFSR could be in use for two-pattern generation. The hardware implementation of this requires counter and control module. This can be given with the equation as

$$HO(n) = 2n \times OR + Control \quad \ldots (3)$$

The hardware overhead of the Carry rotate accumulation based design can be shown in mathematical form as

$$HO(n) = 2n \times DFF + Control \quad \ldots (4)$$

ABAS[11] architecture uses control module independent of the pattern width[10]. The area overhead remains constant for all pattern generations. The mathematical model can be given as

$$HO(n) = 2 \times DFF + Control \quad \ldots (5)$$

The proposed algorithm uses lesser extra hardware units than all the methods discussed above. The design consists of accumulators driven by output of gray code converter or counter. Assuming all the arithmetic modules as included in the design itself, the hardware overhead of the algorithm can be given as

$$HO(n) = 1 \times DFF + Control \quad \ldots (6)$$

The comparisons of hardware of different two pattern schemes are given in Table 2. The comparison is not made on the assumption that, *2*-input NOR is gate equivalent to one gate, one flip-flop as eight gate equivalent and *2*-input XOR gates as 4 gate equivalent.

Table 2—Comparison of two-pattern generation schemes

Pattern Generation Design	*Existing Modules*	*Hardware overhead*
MISR+Cnt	Reg	12*n*
	MISR	8*n*
LFSR	Reg	8*n*
CA	Reg	20*n*
Acc+LFSR	Acc, reg	2*n*+28
Acc+LFSR	Acc,reg	2*n*+18
Acc+ Cntr	Acc,reg	24
Proposed (Gray converter + Acc)	Acc, gray converter Reg	18

Results and Discussion

One of the main reasons behind using arithmetic circuits for pattern generation instead of other generators is reduced area since almost all SoC contains in-built arithmetic circuits. Accumulator based pattern generation scheme uses circuits within the design itself for pattern generation. The area and power of the control module is compared with published results. Table 3 shows the percentage area and power reduction of the proposed design. A test pattern generator with width of 4,8,16 and 32 is used for evaluation. First column of Table 3presents the width of test vector for the experiment; columns second through seven provide the power (in *watts*) and area (in μm^2) of the existing accumulator, arithmetic and proposed accumulator based techniques respectively. Eighth and ninth columns present the reduction in the *Power* × *Area* metric achieved by the proposed scheme. From Table 3, it is observed that the area increases linearly with the width of the test pattern generator, whereas for the case of the proposed scheme both area and power remains the same irrespective of the number of the stages. Hence, the *Power* × *Area* reduction is increasing with the number of the generator stages from 77.57% (for *N*=4) to 85.7% (for *N*=32) in case of APET architecture. The *Power* × *Area* reduction is constant for ABAS architecture[10], the value being 59.6%.

Table 3—Comparison of Area-overhead and Power dissipation of the proposed work with others

Size of TPG (N)	APET[10]		ABAS[11]		Proposed Accumulator based technique		% of Power × Area reduction w.r.to APET	% of Power× Area reduction w.r.to ABAS
	Power ($\times 10^{-4}$)W	Area (μm^2)	Power ($\times 10^{-4}$) W	Area (μm^2)	Power ($\times 10^{-4}$) W	Area (μm^2)		
4	1.841	504.1	1.381	349.6	0.991	211	77.57	59.6
8	1.843	565.1	1.381	349.6	0.991	211	80.1	59.6
16	1.843	687.1	1.381	349.6	0.991	211	83.5	59.6
32	1.846	951.3	1.381	349.6	0.991	211	85.7	59.6

Conclusion

Accumulator based two pattern generation scheme is widely acceptable because it provides higher fault coverage by generating efficient test patterns. This paper suggests a two-pattern generation scheme using accumulators and gray code converters. The single bit changing property of gray code is used here for reduced transition power. The area ×power overhead is lowest for this design compared to any other technique presented in literature. Also it has the advantage of using binary adders which is fastest compared to any other adders.

References

1 Voyiatzis I, Gizopoulos D & Paschalis A, Recursive pseudo-exhaustive two-pattern generation, *IEEE Trans VLSI Sys*, **18(1)** (2010) 142-152.

2 Li N & Dubrova E, Area-efficient high-coverage LBIST, *Micropro and Microsys*, **38(5)** (2014) 368-374.

3 John R T, Sreekanth K D & Sivanantham S, Adaptive Low Power RTPG for BIST based test applications, *Int Conf on Info Comm and Emb Sys,* 21-22 February (2013).

4 Sivanantham S, Gopakumar G, Pandey A & Paikada M J, Adaptive test clock scheme for low transition LFSR and external scan based testing, *3rd Int Conf on Comp Comm and Info*, 4-6 January (2013).

5 Sivanantham S, Mallick P S & Perinbam J R, Low-power selective pattern compression for scan-based test applications, *Comp & Elec Engg*, **40(4)** (2014) 1053-63

6 Sivanantham S, Manuel J P, Sarathkumar K, Mallick P S & Perinbam J R, Reduction of test power and test data volume by power aware compression scheme, *Int Conf Advan in Comp and Comm (ICACC),* Aug 9 (2012) 158-161.

7 Nan-Cheng Lai & Sying-Jyan Wang, On-chip test generation mechanism for scan-based two-pattern tests, *Int Asian Test Symp (Sapporo)*, Nov (2008) 251-256.

8 Voyiatzis I, Efstathiou C, Antonopoulou H & Milidonis A, An effective two-pattern test generator for Arithmetic BIST, *Comp & Elect Engg*, **39(2)** (2013) 398-409.

9 Chen Ch & Gupta, S, BIST test pattern generators for two-pattern testing-theory and design algorithms, *IEEE Trans Comp*, **45(3)** (1996) 257–269.

10 Voyiatzi I, Efstathio C, Antonopoulou H & Milidonis A, Arithmetic module-based built-in self- test architecture for two-pattern testing, *IET Comp and Dig Tech*, **6** (2012) 195-204.

11 Voyiatzis I, Accumulator-based pseudo-exhaustive two-pattern generation, *J of Sys Archi*, **53 (11) (**2007) 846-860.