Accumulator-based Generation for Serial TPG

Ioannis Voyiatzis, Costas Efstathiou
Department of Informatics, Technological Educational Institute of Athens, Greece

Abstract
Arithmetic modules can be utilized to generate test patterns and compact test responses. In this work we investigate the utilization of an accumulator whose inputs are driven by a Feedback Shift Register as a candidate structure for bit-serial test-pattern generation. The proposed scheme compares favorably to previously proposed schemes with respect to the length of the output sequence / hardware overhead tradeoff.

1. INTRODUCTION

Built-In Self-Test (BIST) [1]–[4] schemes reduce the need for external testing equipment and can be used to determine faulty parts during manufacturing and in the field. In BIST, pattern generation and response evaluation are handled on chip with the use of hardware structures. Arithmetic BIST [5]–[21] is based on the use of adders, subtractors, multipliers, and shifter modules that already exist in modern general-purpose processors and digital signal processing units. The advantage of arithmetic BIST is that due to the reuse of existing on-chip modules, hardware overhead and performance degradation are reduced.

Test-per-scan BIST is employed in sequential circuits with scan paths, embedded cores with isolation ring, circuits with boundary scan and portions of multichip modules, which require the transfer of test data in a bit-serial way. The period of the bit-serial test sequence generated by any bit position of an accumulator accumulating a constant value is low [11], therefore, simple arithmetic units are not efficient for bit-serial test-pattern generation. Bit-serial test-pattern generation schemes based on the use of adder–multiplier or accumulator–multiplier pairs were proposed in [10]. These schemes achieve similar fault coverage with similar number of test patterns compared to Linear Feedback Shift Registers (LFSR) bit-serial test-pattern generators. However, their applicability is limited to cases in which the required configuration of the adder–multiplier or accumulator–multiplier is available.

In [19] the modification of accumulators to operate in test mode as nonlinear feedback shift registers was presented. The quality of this scheme depends on the selection of a constant additive value. However, finding an additive value to guarantee maximum sequences requires exhaustive searching. Furthermore, for some accumulator sizes, no additive value that ensures maximum period of bit sequences was found. In [20] another accumulator-based configuration is proposed that generates a sequence with period 2^k-1 for all values of k, the accumulator width.

In this paper we propose an alternative bit-serial accumulator-based scheme where the inputs of the accumulator are driven by the outputs of an easily implemented feedback shift register. Comparisons with the schemes in [19], [20] indicate that the proposed here scheme results in lower hardware overhead for the same length of the resulting sequence.

The paper is organized as follows. In Section 2 some theoretical background is presented. In Section 3 the proposed scheme is implemented and illustrated. In Section 4 the scheme is compared with the previously proposed ones; in Section 5 we conclude the work. Table 2 and Figure 3 can be found at the end of the manuscript for layout reasons.

2. BACKGROUND

The analysis of this Section is similar to the one investigated in [21] in a different context. First we note that if N and k are non-negative integer numbers, then the following relation holds (GCD and LCM denote the Greatest Common Divisor and Least Common Multiple, respectively).

$$N \times k = LCM(N, k) \times GCD(N, k) \Rightarrow \frac{LCM(N,k)}{k} = \frac{N}{GCD(N,k)}$$

In the sequel, we shall denote with n the number of bits in the number; $N=2^n$, i.e. N is a power of 2. Hence, the only numbers dividing N are 2^i, $0<i\leq n$. Thus GCD(N, k)>1 if and only if k is even, and GCD(N, k)=1 if and only if k is odd.

Lemma 1: If we start from any binary value A and consecutively add a constant value k (modulo N) we shall return to A in $\frac{N}{GCD(N,k)}$ cycles.

Proof: Suppose we return to A after m STEPS. Then
$A + k\times m \equiv A \bmod N \Rightarrow k\times m \equiv 0 \bmod N \Rightarrow N\times c = k\times m = i$
where c, i are integers. The smallest integer i satisfying the above relation is LCM(N,k). Thus,
$m = \frac{LCM(N,k)}{k} = \frac{N}{GCD(N,k)}$, Q.E.D.

Corollary 1: If we consecutively add an odd number modulo N, then we will return to the initial value after N steps.

Corollary 1 states the well-known fact that if we repeatedly accumulate an n-bit odd number in an n-stage accumulator we will generate all n-bit patterns before returning to the initial value.

Definition 1: A Sequence(k) is a sequence of all integer numbers from 1 to k, where each number is taken exactly once.

Note that for k>1 more than one Sequences(k) exist (in fact, the number of Sequences(k) is $1\times2\times3..\times k=k!$) For example,

PCI 2015, October 01-03, 2015, Athens, Greece
© 2015 ACM. ISBN 978-1-4503-3551-5/15/10…$15.00
DOI: http://dx.doi.org/10.1145/2801948.2801969

the Sequences(3) are {1,2,3}, {1,3,2}, {2,1,3}, {2,3,1}, {3,1,2}, {3,2,1}.

In the sequel we shall denote by S_k the sum of the values of the elements of a Sequence(k). S_k is irrespective of the order of the patterns in the Sequence(k) and is equal to k×(k+1)/2.

Definition 2: A Circle(k, N, A) is a sequence of vectors generated starting from A and consecutively accumulating (modulo N) the elements of a Sequence(k) until we return to A after accumulating the last element of the Sequence(k).

According to the definition of the Circle, there are k! different Circles(k, N, A) starting from the same starting value A, one for each utilized Sequence(k). For example, let n=2 (N=4) and k=2. The Sequences(2) are {1,2} and {2,1}. The Circles(2, 4, 0) that correspond to these sequences are presented below:

Sequence(2)={1,2}

	A	1	2	1	2	1	2	1	2
Circle(2,4,0)	0	1	3	0	2	3	1	2	0

Sequence(2)={2,1}

	A	2	1	2	1	2	1	2	1
Circle(2,4,0)	0	2	3	1	2	0	1	3	0

Theorem 1: If S_k is odd, a Circle(k, N, A) will return to A (the initial value) after k×N steps,.

Proof: Let us consider an inclusive step stating when a Sequence starts and completing when a Sequence completes. From Lemma 1, is Sk is odd then N Sequences(k) will be applied. Hence, since each Sequence generates k steps, the number of applied steps is k×N. Q.E.D.

Corollary 2: A Circle(N-2, N, 0) generates (N-2)×N patterns before returning to the initial state.

Proof: S_{N-2} is an odd number, since $S_{N-2} = \frac{(N-1)\times(N-2)}{2} = \frac{(2^n-1)\times(2^n-2)}{2} = (2^n-1)\times(2^{n-1}-1)$ which is odd as the product of two odd numbers. Therefore the proof follows directly from Theorem 1. Q.E.D.

For example, let us consider the case of a three bit generator. The possible values that can be generated by an LFSR implementing a primitive polynomial are (in decimal) 1, 2, 3, 4, 5, 6 and 7 in different arrangements. According to the above discussion, if the sequence 1, 2, 3, 4, 5, 6 is repeatedly applied to the inputs of a 3-stage accumulator, then a sequence consisting of $(2^n-2)\times 2^n = 6\times 8 = 48$ patterns is generated. Therefore, the $(2^n-2)\times 2^n$ sequence is non-repeated. Hence, without need for extra control, the sequence generated is distinct for these $2^n\times(2^n-2)$ patterns.

After the above discussion, in order to generate a sequence with period $(2^n-2)\times 2^n = (N-2)\times N$, we can feed an accumulator with a Sequence(N-2). In the next Section we shall present an implementation of the scheme along with the design of a Non-linear Feedback Shift Register (NFSR) to generate the Sequence(N-2). The design of the NFSR stems from Theorem 2.

Theorem 2: If an n-stage maximal cycle external LFSR is initialized to the pattern 011...1 and clocked until the pattern 11...10 is generated, then a Sequence(N-2) is generated at the outputs of the LFSR.

Proof: First, note that in the sequence of vectors generated by an external maximal cycle LFSR, if {11...1} is generated at cycle number t, then
(a) {11...10} is generated at cycle t-1 (i.e. in the previous cycle) and
(b) {011...1} is generated at cycle t+1 (i.e. in the next cycle).

Indeed, let P denote the vector generated at cycle t-1 and S denote the vector generated at cycle t+1. The n-1 high-order bits of P are 1, since they are shifted to the right (in the next cycle) to form the n-1 low-order bits of {11...1}; the high-order bit is 0 (since if it were 1, then no change would occur in the LFSR); thereby, P={111...0}.

Similarly, the n-1 low-order bits of S are 1, since they were (in the previous cycle) the n-1 high-order bits of {11...1}, which are shifted to the right. The high-order bit of S is 0 (since if it were 1, then no change in the state of the LFSR would occur); thereby, S={011...1}.

From the above it is implied that if the LFSR is initialized to S and clocked until P is generated, N-2 non-zero n-bit patterns are generated (all non-zero patterns except from {11...1}). Therefore, N-2 distinct vectors are generated, whose decimal values are from 1 to N-2. Q.E.D.

According to Theorem 2, if an LFSR is initialized to 011...1 ($=\frac{N}{2}$-1 decimal) and clocked for N-2 cycles, all non-zero vectors except 11...1 are generated. This will be exploited in the next Section for the design of the NFSR structure.

3. Implementation

Following the results of the previous Section, the implementation is based on a binary accumulator whose inputs are driven by a Non-Linear Feedback Shift Register (NFSR) than can generate the Sequence(N-2), as shown in Figure 1.

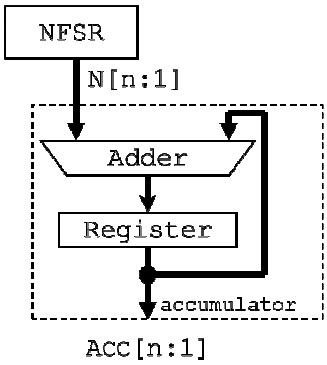

Figure 1: NFSR-based hardware implementation

For the implementation of the NFSR that generates the Sequences(N-2), the results of Theorem 2 are used. More precisely, what is needed is a maximal cycle LFSR modified in such way that it cycles through all the patterns except from the all-1 pattern. This can be achieved by inserting gates with (n+1) inputs in total.

For example, let us consider the 3-stage maximal cycle LFSR presented in Figure 2(a). The NFSR of Figure 2(b) stems from the LFSR of Figure 2(a) with the addition of gates with totally 5 inputs. If the structure of Figure 2(b) is initialized to the pattern 011, the following sequence of vectors is repeatedly generated: N[3:1]: {011, 101, 010, 001, 100, 110}, i.e. the Sequence(6)={3,5,2,1,4,6}.

In Table 1 we present the sequence generated from a 3-stage accumulator whose inputs are driven from the 3-stage NFSR of Figure 2(b). In Table 1 in the first column we present the cycle number; in the second column we present the value added to the input of the accumulator; in the 3rd, 4th, 5th column we present the values of the output bits of the accumulator. From Table 1 it is trivial to note that the period of the leftmost bit is 48 (after cycle #48, the patterns 6 will be accumulated, that will return the accumulator to state 000 and the sequence will start from the beginning).

Table 1: Sequence generated by applying the sequence {3, 5, 2, 1, 4, 6} in a 3-stage accumulator

#	Added value	#3	#2	#1	#	Added value	#3	#2	#1
1		0	0	0	25	6	1	0	0
2	3	0	1	1	26	3	1	1	1
3	5	0	0	0	27	5	1	0	0
4	2	0	1	0	28	2	1	1	0
5	1	0	1	1	29	1	1	1	1
6	4	1	1	1	30	4	0	1	1
7	6	1	0	1	31	6	0	0	1
8	3	0	0	0	32	3	1	0	0
9	5	1	0	1	33	5	0	0	1
10	2	1	1	1	34	2	0	1	1
11	1	0	0	0	35	1	1	0	0
12	4	1	0	0	36	4	0	0	0
13	6	0	1	0	37	6	1	1	0
14	3	1	0	1	38	3	0	0	1
15	5	0	1	0	39	5	1	1	0
16	2	1	0	0	40	2	0	0	0
17	1	1	0	1	41	1	0	0	1
18	4	0	0	1	42	4	1	0	1
19	6	1	1	1	43	6	0	1	1
20	3	0	1	0	44	3	1	1	0
21	5	1	1	1	45	5	0	1	1
22	2	0	0	1	46	2	1	0	1
23	1	0	1	0	47	1	1	1	0
24	4	1	1	0	48	4	0	1	0

4. COMPARISONS

In order to investigate the merit of the proposed scheme we compare it with previously proposed accumulator-based serial sequence generators that have been proposed in the literature [19], [20]. These schemes utilize a k-stage accumulator comprising a properly modified adder and a register; the modifications of the adder consists of a series of n 2-to-1 multiplexers, and one XOR gate to generate a sequence with length. Therefore, assuming the availability of an k-stage accumulator and considering that a multiplexer requires 1.7 gates, the hardware overhead is $1.7\times k+1.3$ (for [19]) and $1.7\times k + 3.3$ (for [20]); the resulting schemes can generate a sequence with length 2^k-1.

For the implementation of the proposed scheme, considering the availability of an n-stage accumulator and the register that will form the NFSR to feed the accumulator inputs, the hardware overhead will be (n+1) input gates plus one XOR gate to form the linear feedback. Therefore (we consider a XOR gate requires 1.7 gates as is the case for the multiplexer) the overhead is (n+1)+1.7 = n+2.7; the length of the resulting sequence is $2^n\times(2^n-2)$.

It should be noted that the values of k (for the schemes in [19], [20]) and n (for the proposed scheme) are different. This is due to the fact that the proposed scheme generates sequences of the order $2^n\times(2^n-2)$, while the schemes in [19], [20] generate patterns of the order 2^k-1.

Table 2 summarizes, for various values of the length of the sequence, the hardware overhead of the three schemes. In Table 2, in the first column we present the number of the stages of the modules for [19] and [20]. In the second and fourth column we present the length of the generated sequence; in the third and fifth column we present their respective hardware overhead. In the sixth through the eighth column we present the respective data for the proposed scheme. Finally, in the ninth column we present the decrease in hardware presented by the proposed scheme.

From Table 2, the value of n is roughly half of k, since they result in test lengths of the same order.

Figure 3 graphically illustrates the hardware overhead of the various schemes as a function of the binary logarithm of L, the length of the required sequence.

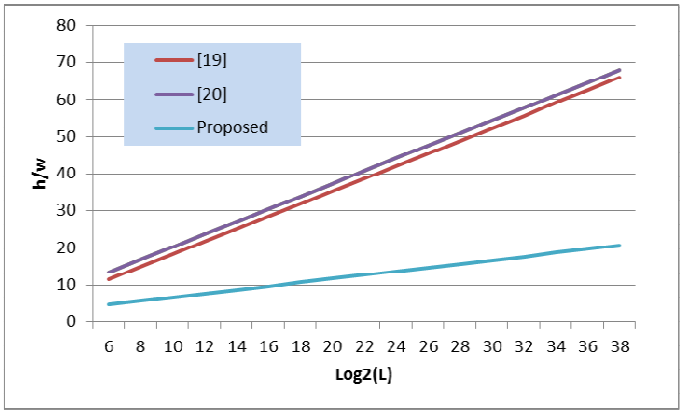

Figure 3: Accumulator-based bit-serial TPG schemes: comparison

It should be also noted that, since the proposed scheme is based on a smaller data path (roughly half the size of the previously proposed schemes) the maximum operation frequency is expected to be higher. Furthermore, in case a large datapath is not available the proposed scheme is further favoured. For example, if a length of 4×10^{10} is required, then the previously proposed schemes require a 32-bit datapath, while the proposed scheme can generate the same length with a 16-bit datapath.

5. CONCLUSIONS

We have proposed the utilization of binary accumulators whose inputs are fed by the output of an NFSR structure, in order to generate bit serial patterns. Comparisons with previously proposed accumulator-based schemes indicate that

the proposed here scheme results in considerably lower hardware overhead for the same length of the generated sequence.

ACKNOWLEDGMENT

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Archimedes III, Investing in knowledge society through the European Social Fund.

References

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman, Digital Systems Testing and Testable Design. New York: Computer Science, 1990.

[2] P. H. Bardell, W. H. McAnney, and J. Savir, Built-In Test for VLSI:Pseudo-Random Techniques. New York: Wiley, 1987.

[3] M. Bushnell and V. Agrawal, Essentials of Electronic Testing for Digital, Memory & MixedSignal VLSI Circuits. Boston, MA: Kluwer, 2000.

[4] H. J. Wunderlich, "BIST for systems-on-a-chip," Integr. VLSI J., vol. 26, no. 1/2, pp. 55–78, Dec. 1998.

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

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

[7] A. P. Stroele, "Test response compaction using arithmetic functions," in Proc. 14th IEEE VLSI Test Symp., Princeton, NJ, Apr./May 1996, pp. 380–386.

[8] S. Gupta, J. Rajski, and J. Tyszer, "Arithmetic additive generators of pseudo-exhaustive test patterns," IEEE Trans. Comput., vol. 45, no. 8, pp. 939–949, Aug. 1996.

[9] A. P. Stroele, "BIST pattern generators using addition and subtraction operations," J. Electron. Test.—Theory Appl., vol. 11, no. 1, pp. 69–80, Aug. 1997.

[10] J. Rajski and J. Tyszer, "Multiplicative window generators of pseudorandom test vectors," in Proc. Eur. Design and T est Conf., Paris, France, Mar. 1996, pp. 42–48.

[11] A. P. Stroele, "Bit serial pattern generation and response compaction using arithmetic functions," in Proc. IEEE VLSI Test Symp., Monterey, CA, Apr. 1998, pp. 78–84.

[12] I. Voyiatzis, "An Accumulator - based compaction scheme with reduced aliasing for on-line BIST of RAMs", IEEE Transactions on VLSI Systems, vo. 16, no. 9, September 2008, pp. 1248-1251.

[13] S. Chiusano, S. Di Carlo, P. Prinetto, and H. J. Wunderlich, "On applying the set covering model to reseeding," in Proc. Design, Automation Test Eur. Conf., Munich, Germany, Mar. 2001, pp. 156–160.

[14] D. Magos, I. Voyiatzis, S. Tarnick, "An Accumulator-Based Test-per-clock Scheme", IEEE Transactions on VLSI Systems, (Volume:19 , Issue: 6), June 2011, Page(s): 1090 - 1094.

[15] I. Voyiatzis, "An ALU based BIST scheme for Word-organized RAMs", IEEE Transactions on Computers, vol. 57, no. 58, May 2008, pp. 577-590.

[16] S. Manich, L. Garcia, L. Balado, E. Lupon, J. Rius, R. Rodriguez, and J. Figueras, "On the selection of efficient arithmetic additive test pattern generators," in Proc. IEEE Eur. Test Workshop, Maastricht, The Netherlands, May 2003, pp. 9–14.

[17] I. Voyiatzis, "Test vector embedding into accumulator-generated sequences: A linear-time solution," IEEE Trans. Comput., vol. 54, no. 4, pp. 476–484, Apr. 2005.

[18] I. Voyiatzis, "Aliasing Reduction in Accumulator-based Response Verification", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on (Volume: 33 , Issue: 11), Page(s): 1746 – 1750.

[19] G. Dimitrakopoulos, D. Nikolos, D. Bakalis, "Bit-Serial Test Pattern Generation by an Accumulator Behaving as a Non-Linear Feedback Shift Register", 8th IEEE International On-Line Testing Workshop, Isle of Bendor, France, July 8-10, 2002

[20] D. Kagaris, P. Karpodinis, D. Nikolos: On Obtaining Maximum-Length Sequences for Accumulator-Based Serial TPG. IEEE Trans. on CAD of Integrated Circuits and Systems 25(11): 2578-2586 (2006)

[21] I. Voyiatzis, A. Paschalis, D. Nikolos, C. Halatsis, "Accumulator-based BIST Approach for two-pattern testing", Journal of Electronic Testing: Theory and Applications, Volume 15, Issue 3 (December 1999), pp. 267 - 278.

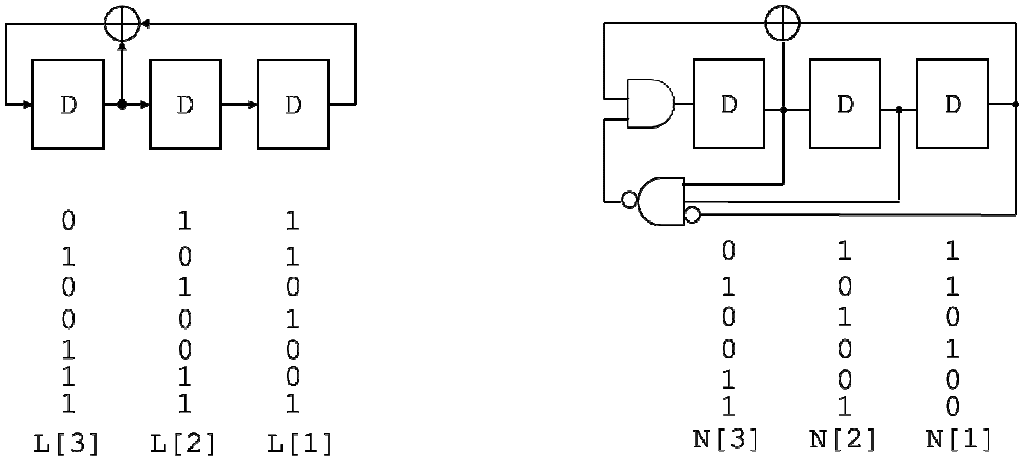

Figure 2: 3-stage LFSR and corresponding NFSR

Table 2: Comparisons

	[19]		[20]		Proposed			
k	**Length**	**h/w**	**Length**	**h/w**	**n**	**Length**	**h/w**	**%Decr**
6	63	12	63	14	3	48	6	65%
8	255	15	255	17	4	224	7	66%
10	1.023	18	1.023	20	5	960	8	67%
12	4.095	22	4.095	24	6	3.968	9	68%
14	16.383	25	16.383	27	7	16.128	10	68%
16	65.535	29	65.535	31	8	65.024	11	68%
18	262.143	32	262.143	34	9	261.120	12	68%
20	1.048.575	35	1.048.575	37	10	1.046.528	13	69%
22	4.194.303	39	4.194.303	41	11	4.190.208	14	69%
24	16.777.215	42	16.777.215	44	12	16.769.024	15	69%
26	67.108.863	46	67.108.863	48	13	67.092.480	16	69%
28	268.435.455	49	268.435.455	51	14	268.402.688	17	69%
30	1073.741.823	52	1.073.741.823	54	15	1.073.676.288	18	69%
32	4.294.967.295	56	4.294.967.295	58	16	4.294.836.224	19	69%
34	17.179.869.183	59	17.179.869.183	61	17	17.179.607.040	20	69%
36	68.719.476.735	63	68.719.476.735	65	18	68.718.952.448	21	69%
38	274.877906.943	66	274.877.906.943	68	19	274.876.858.368	22	70%