Bit-Serial Test Pattern Generation by an Accumulator behaving as a Non-Linear Feedback Shift Register

G. Dimitrakopoulos[†], D. Nikolos[†‡] and D. Bakalis[†‡]
[†]*Computer Engineering and Informatics Dept., University of Patras, 26 500, Patras, Greece*
[‡]*Computer Technology Institute, 61 Riga Feraiou Str., 26 221 Patras, Greece*
e-mails: dimitrak@ceid.upatras.gr, nikolosd@cti.gr, bakalis@cti.gr

Abstract

Arithmetic function modules which are available in many circuits can be utilized to generate test patterns and compact test responses. Recently, it was shown that an adder or an accumulator cannot be used as a bit serial test pattern generator due to the poor random properties of the generated sequences. Thus, accumulator-multiplier or adder-multiplier structures have been proposed. In this paper we show that an accumulator behaving, in test mode, as a Non-Linear Feedback Shift Register (NLFSR) can be used efficiently for bit serial test pattern generation. A hardware as well as a software implementation of the proposed scheme is given. The efficiency of the proposed scheme is verified by comparing it against LFSR and other arithmetic function based bit serial test pattern generators.

1. Introduction

Built–In Self–Test (BIST) [1-4] technique gained increasing interest in the past few years as it provides with little cost, a well-defined increase in the testability of the Circuit Under Test (CUT) offering at the same time a structured and modular approach in the problem of testing a digital system from board-level down to single-chips. In BIST, test pattern generation and response monitoring and evaluation are handled on-chip, with the use of extra hardware structures. Common BIST schemes used in practice for many years are based on the use of Linear Feedback Shift Registers (LFSRs) or cellular automata for test pattern generation and response compaction. Such conventional approaches impose hardware overhead and may lead to performance degradation, during normal operation mode, due to the insertion of extra multiplexers in the signal paths. Recently, new Arithmetic-BIST [5-13] schemes were proposed based on the use of adders, subtracters, multipliers and shifter modules that already exist in modern general purpose processors and digital signal processing units. The advantage of Arithmetic-BIST against the LFSR-based BIST is that due to the reuse of existing on-chip modules hardware overhead and performance degradation are reduced or virtually eliminated.

Arithmetic BIST schemes for test-per-clock as well as for test-per-scan environment have been considered [6-13]. In this paper we concentrate on the arithmetic test-per-scan BIST environment. In this case, test patterns are generated in a bit-per-bit fashion and shifted along the primary inputs and the scan path of the CUT, while the collected responses are shifted out in a similar way, in order to be evaluated. Common cases where such an approach is compulsory are. a) Sequential circuits with scan paths, b) Embedded cores with an isolation ring, c) Circuits with a boundary scan path and d) Portions of multi-chip modules, which require the transfer of test data in a bit-serial way.

The quality of the random properties of the bit serial test sequence generated by a bit position of an accumulator or an adder is poor [11]. Therefore these simple arithmetic units cannot be used efficiently for bit serial test pattern generation. To this end, three new bit serial test pattern generation schemes based on the use of adder-multiplier or accumulator-multiplier pairs were recently proposed in [10, 11]. These schemes compared to an LFSR bit serial test pattern generator have the advantage that achieve similar fault coverage with similar number of test patterns, while they do not impose any hardware overhead since they are already part of the functional circuit. The disadvantages of the schemes proposed in [10,11] are. a) Their applicability is limited to applications in which the required configuration of the adder-multiplier or accumulator-multiplier is available and b) since a multiplier-adder or multiplier-accumulator is used for test pattern generation, these schemes have increased power and energy consumption during testing.

Recently in [12] it was shown that an accumulator can be modified to operate, in test mode, as a Non-Linear Feedback Shift Register (NLFSR) and that it can be used effectively for test response compaction [12, 13]. In this paper we show that an accumulator behaving as a NLFSR can also be used as a bit-serial test pattern generator achieving the same fault coverage with the schemes

This work was partially supported by GiGA Hellas S.A. an Intel Company.

proposed in [10, 11] using in most cases a smaller number of test vectors. Furthermore, the applicability of the proposed scheme is wider, since, in contrast to the schemes proposed in [10, 11], it does not require the availability of a multiplier in the functional circuit.

The rest of the paper is organized as follows. Section 2 refers to bit serial test pattern generation schemes based on arithmetic units. In Section 3 we present a new accumulator-based bit serial test pattern generation scheme, while on Section 4 we give experimental results in order the proposed scheme to be evaluated and compared with the already known bit-serial test pattern generation schemes.

2. Bit-Serial Arithmetic Test Pattern Generation: A Retrospection

Bit-serial test pattern generation based on arithmetic units was firstly investigated in [10]. In particular Rajski and Tysjer [10] presented a datapath configuration which consists of a $k\times k$ multiplier, a $2k\times 2k$ adder and $2k$ register (see Figure 1.a).

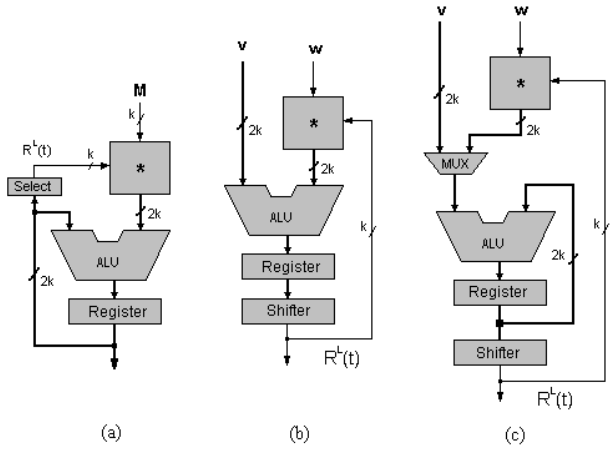

Figure 1. The multiplier structures presented in [10, 11] for bit-serial test pattern generation.

In each step the k lower significant bits of the register are multiplied with a constant M and the product is added to the rest k most significant bits of the register. The state transitions are described by, $R(t) = [M\cdot R^{(L)}(t\text{-}1) + R^{(H)}(t\text{-}1)] \bmod 2^{2k}$, where $R(t)$ denotes the contents of the register, while $R^{(L)}(t)$ and $R^{(H)}(t)$ are the k less significant and the k most significant bits of the register respectively. The bit-sequence that feeds the scan path is generated by the least significant bit position of the register. The constant M, which guarantee a maximal period, have been derived experimentally for k = 3 up to 16 along with the corresponding period for each k.

All application specific circuits are tailored to and optimized for specific tasks. Hence, their busses do not connect registers and functional units in a completely regular fashion. In order to cope with the variety of hardware structures Stroele [11] asserts that we need a set of many different pattern generators. Then we will be able to choose those pattern generators that can be easily configured from the available arithmetic function units and are best suited for the specific situation. To this end Stroele [11] investigated the suitability of several simple and more complex arithmetic modules for bit-serial test pattern generation.

Stroele [11] shown that 2's complement adders, 1's complement adders as well as stored overflow bit adders are not suitable for bit-serial test pattern generation due to the poor random quality of the sequences generated from any bit position. Stroele [11] has also investigated the capabilities of two additional schemes called Multiply & Add and Multiply & Accumulate respectively (Figure 1.b).

The first one consists of a $k\times k$ multiplier, a $2k\times 2k$ adder and register, while the design is completed with the use of an extra shifter which selects in every step the k-lower significant bits of the register. The state transitions are described by $R(t) = [w\cdot R^{(L)}(t\text{-}1) + v] \bmod 2^{2k}$, which means that in each time step the lower significant bits of the register are multiplied with a constant w and the product is added to v. As presented in [11] the sequence generated by the multiply-add configuration consists of a long cycle passing through 2^k states, when the constant additive value v is odd and the constant w is an element of $\{4x+1 \mid x \in N\}$. The source of the serial random vectors is the bit position R_{k-1} which generates bit-sequences with period 2^k.

The second structure needs an extra $2k$ bits wide 2-to-1 multiplexer and a new pattern is produced in every two cycles. During the first cycle the k-lower significant bits of the register $R^{(L)}(t)$ are multiplied with the constant w and the product is added to the contents of the register. In the second cycle the constant additive value v is also added to the register's value. The presented function is described by the recursive equations, $R(t+1)=[R(t)+w\cdot R^{(L)}(t)] \bmod 2^{2k}$ and $R(t+2) = [R(t+1) + v] \bmod 2^{2k}$. The Multiply-Accumulate configuration generates random patterns with period 2^{2k} and the sequence produced by the most significant bit of the k lower significant bits of the register has exactly the same period.

The above mentioned multiplier-adder and multiplier-accumulator based schemes exhibit attractive pseudo-random and fault-detecting characteristics that are sometimes even better than those of the conventional LFSRs, achieving high-fault coverage with a rather small number of test vectors. We have to note that any of the schemes proposed in [10,11] can also be implemented in software in a microprocessor environment.

3. The Proposed Scheme

The basic module of the proposed test pattern generation scheme is the accumulator consisting of an adder and a k-bits register. The structure is completed with the addition of k 2-to-1 multiplexers $M_1, M_2, \ldots, M_k$, a XOR

gate and an extra flip-flop (D-FF) as shown in Figure 2.

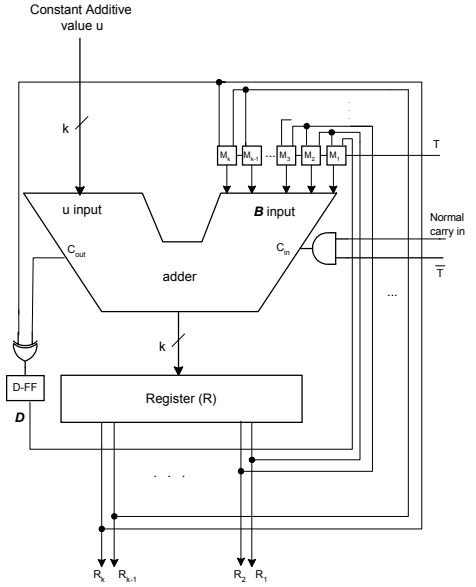

Figure 2. The proposed Accumulator behaving in test mode as a NLFSR.

In normal mode, T=0, the output R_i, $1 \le i < k$, of the register drives the B_i input of the adder, while the exclusive-OR gate and the D-FF flip-flop are not used. In test mode, T=1, the output R_i, $1 \le i < k$, of the register drives the B_{i+1} input of the adder and the output R_k feeds the additional exclusive-OR gate. Additionally, the output of the D-FF flip-flop drives the B_1 input of the adder while C_{in} receives the value zero. Therefore in test mode the modified accumulator of Figure 2 behaves identically to the circuit of Figure 3 which is a NLFSR [12]. Functionally we can say that in test mode the contents of the register are shifted one position to the left and are then added with a constant value *u* and the value *D* of the *D-FF* flip-flop. We store the *k* bits of the result back to the register and the (*k*+1)-bit at the *D-FF*.

The hardware overhead calculation will be based on gate equivalents. Using the Synopsys tools driven by the AMS CUB implementation technology (0.6μm, 2-metal layer, 5.0V) and taking one 2-input NAND or one 2-input NOR gate equal to 1 gate equivalent, we get: one 2-input exclusive-OR gate equals 2.0 gate equivalents, one 2-input AND gate equals 1.3 gate equivalents, one D flip-flop equals 3.6 gate equivalents and a 2-to-1 multiplexer equals 1.7 gate equivalents. Then the hardware required so that an accumulator to behave, in test mode, as a NLFSR is equal to *1.7k+6.5* equivalent gates. The hardware overhead of the MAC [10], Multiply-Add and Multiply-Accumulate schemes [11] is zero. However, they assume the existence of a multiplier-accumulator or multiplier-adder pair in the circuit. According to our scheme only the existence of an accumulator is required. In most LFSR based test-per-scan BIST, the LFSR is a dedicated circuit. In this case the hardware required for the implementation is equal to *3.6k+2m* where *m* is the number of the XOR gates. We can see that the hardware overhead imposed by modifying the accumulator is roughly equal to the half of the cost of the LFSR. In the case that an existing register is modified to function in test mode as a LFSR then the cost is equal to *1.7k+2m* equivalent gates, which is similar to the cost of our scheme.

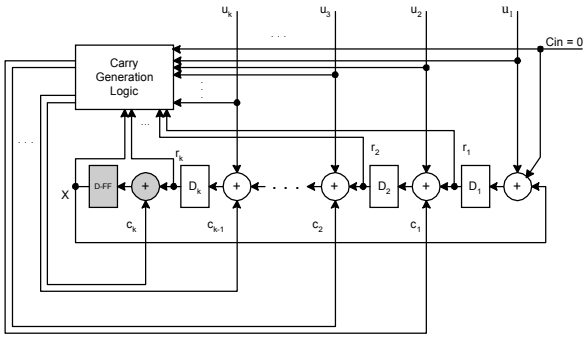

Figure 3. The accumulator behaving as a Multiple Input Non-Linear Feedback Shift Register

In the case that the accumulator belongs to the datapath of a processor then, instead of the hardware implementation, the NLFSR can be emulated by a program. In this case the hardware overhead is zero. A general form of code that can be used to generate one test bit is shown on Figure 5. At each step, the state *s(t)* is stored in the accumulator (ACC) and in register R_{flag} which emulates the D-FF added in the hardware version of the NLFSR. Lines 1-4 produce the next state in the accumulator while the code in lines 5-7 update the value of the D-FF (R_{flag}) by adding modulo-2 the bit that was shifted out and the carry-out produced from the addition step in lines 3 and 4. In general, the code segment of Figure 4 should be executed N x L times in order to generate L vectors, of length N each.

```
SHL     ACC              % Shift Left Accumulator
ADDC    Temp,#0          % Preserve carry out in temp
ADD     ACC,ACC,u        % Add constant u to ACC
ADDC    ACC,ACC,Rflag,   % Add the contents of Rflag
LD      Rflag,#0         % Clear Rflag
ADDC    Rflag,#0         % Preserve carry out in Rflag
XOR     Rflag,Rflag,Temp % Update Rflag for the next loop
```

Figure 4. The code segment that emulates one state transition of the NLFSR.

Let us now declare *s(t)* the state of the generator of Figure 2 which is defined as the contents of the register *R(t)* and the contents of the extra flip-flop *D(t)*, considering *D(t)* as the most significant bit of *s(t)*. In other words *s(t)* is equivalent to a (*k+1*)-bit vector with elements $\{D(t),R_{k-1}(t),...,R_1(t),R_0(t)\}$. Taking into account the previous description of operation, the next state is computed by the following recursive equation.

$$s(t+1) = \begin{cases} [2s(t)+u] \bmod 2^{k+1}, & s(t) < 2^k \\ [2s(t)+u+1] \bmod 2^{k+1}, & s(t) \ge 2^k \end{cases}$$

The period of the generated sequence strongly depends on the proper selection of the constant additive value *u*. In the rest of the paragraph we will investigate the structural

properties of the generated sequence and how they are affected by the selection of the constant additive value *u*. After performing several experiments and by thoroughly examining the generated sequences we have made the following observations. a) For the majority of different register sizes there is at least one constant additive value *u* that gives a maximum period $p = 2^{k+1}-1$. For these cases the state transition diagram of the proposed scheme consists of a single long cycle of length $2^{k+1}-1$, which implies that the test pattern generation scheme can pass through *p* different states irrespective of the initial state *s(0)*. b) For the cases where a maximal period cannot be obtained such as for $k = 12$, there is a constant additive value that gives a period very close to $2^{k+1}-1$. Hence in Table 2 we present for each *k* the additive values that give the maximum period, along with the obtained period. The cases that the period is equal to $2^{k+1}-1$ have been shaded. c) Stroele has proven [9] that if the transition diagram, of either a k-bit accumulator operating under constant input or an arbitrary finite state machine with k-state bits, contains a cycle of length 2^k-1, then the generated bit sequences from each bit position *i, i=0,1,...,k-1* has period 2^k-1. Therefore, when the proposed test pattern generator, with k+1 state bits, operates under a constant input and produces a state cycle of length $2^{k+1}-1$ (only one state is not reached for every *k*) then the sequence generated from the *j*-th bit position $R_j(t-1)$, $R_j(t)$, $R_j(t+1)$,… has period $p_j = 2^{k+1}-1$ for $j = 0, 1, \ldots, k-1$.

Table 1. The constant additive values that ensure a maximal period for each *k*

K	Constant Additive Value					Period
5	16	20	22	24		63
6	32	34				127
7	60					244
8	168	170				511
9	320					1023
10	834					2038
11	1282	1302	1644	1648	1650	4095
12	2368					8169
13	5216					16368
14	8192 9644 9648	9650 9652 9656	9658 9672 9674	9676 9678 9698	10218 10252 10254	32767
15	24218 24222	24226 24230	24232 24316			65535
16	40864					131071
17	65526					262094
18	142634 142636 142688	142690 142694 142696	142700 142982 142992	143022 143024 143026		524287
19	327258 327260	327262 327264	327754 327982	327984 327994		1048575
20	841936					2097134

Since bit serial test pattern generation is our primary interest the Most Significant Bit of the Register (R_{k-1}) was chosen to be the source of the random test sequence. The selection of the most significant bit position is based upon the observation presented in [14] which states that the least significant bits of a pseudorandom number sequence are much less "random" than the most significant ones. Using as randomness metric the Cesaro method for the calculation of π we found out that the randomness of the proposed test pattern generators approaches the random quality of the sequences generated by the corresponding MAC [10], adder-multiplier, accumulator-multiplier [11] and LFSRs. Furthermore the analysis of the generated sequence with respect to the length of runs of "0" and "1" as well as the number of combinations appearing in a sliding window of four adjacent bits [11] give promising results.

Motivated by Gold Sequences [15] we investigated the random properties of the sequences generated from the output of a XOR gate receiving as inputs the sequences generated by two different bit positions of the register. We examined the sequences XOR (R_{k-1}, R_i), with i=0,1,…,k-2, using as randomness metric the Cesaro method for the calculation of π. In all cases the sequence that appeared to be more random was the one that combined the most significant bits of the upper and the lower part of the register, as shown in Figure 5.

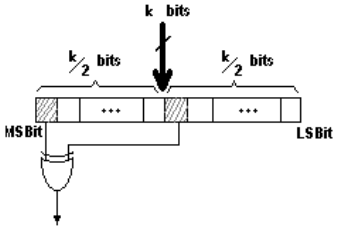

Figure 5. The serial output of the Enhanced test pattern generator

The randomness of the sequence generated from the output of the additional XOR gate is improved against the case that the XOR gate is not used and is similar to the randomness of the sequences generated by the corresponding MAC [10], adder-multiplier, accumulator-multiplier [11] and LFSRs. Also their structural properties have improved compared to the case that the XOR gate is not used. The improvements are valid even in the cases that the constant additive value is chosen randomly. For example, we considered a 24-bit wide accumulator and chose an additive constant value *u=9642306*. The chosen value produces numbers with period 14790812, which is less than the maximum possible period $2^{25}-1$. Producing a 10^6 bit sequence either from the most significant bit position of the register or from the output of the additional XOR gate, all possible runs of "0" and "1" with length up to 17 appeared. A difference appeared when a sliding window of 4 adjacent bits in the sequence was considered. In the first case 9 of all possible 16 combinations occurred while in the second 12, thus improving the structural properties of the sequence.

The quality of the new generated sequence will be far more evident from the results that we present in the next section along with the results obtained from the simple version of the proposed generator, and the already known

bit-serial test pattern generation schemes. In the following sections we will refer to the proposed test pattern generator as "Simple" when the MSB of the register is the source of the test sequence and as "Enhanced" when the test sequence is produced from the output of the additional XOR gate.

4. Evaluation and Comparisons

A lot of experiments were conducted in order to evaluate the quality of the test sequences generated by our scheme. In the first set of our experiments we used the non-redundant version of the ISCAS'85 benchmark circuits [16] while in the second set we present results of the fault coverage obtained using the ISCAS'89 benchmarks [17]. For both benchmarks sets we assume that the primary inputs and the internal flip-flops are connected to a single scan chain. The fault coverage in every case was calculated as the fraction of the number of faults detected by the test vectors of the test pattern generator over the total number of detectable faults. In each case, the number of clock cycles that are used to produce and shift-in a new test vector were chosen to be relatively prime to the period of the generated sequence, in order to guarantee that a maximal number of different patterns can be applied to the CUT.

For each type of pattern generator, 4 different input constants or 4 different primitive polynomials were investigated and 10 randomly selected initial states were tried in each case. Thus, 40 experiments were conducted for each type of pattern generator and for each benchmark circuit respectively. The input constants for the arithmetic bit serial test pattern generation structures where chosen according to the results reported in [10] for the MAC structure and the theorems presented in [11] for the multiply-add and multiply-accumulate schemes. LFSRs have been designed according to the primitive polynomials taken from [2].

Table 2. Results achieved by 16-bit TPGs

CUT	LFSR	[11]		[10] MAC	Proposed	
		Mult& Add	Mult& Acc		Simple	Enhanced
c432	366	99.52%	510	454	425	273
c499	645	97.15%	572	603	491	475
c880	6377	95.97%	99.77%	3985	4750	4096
c1355	1541	94.55%	1426	1529	1421	1400
c1908	4288	86.25%	99.97%	5256	4983	4675
c2670	90.90%	82.55%	84.41%	81.84%	84.36%	90.74%
c3540	12607	91.02%	99.89%	11514	99.92%	10940
c5315	1862	98.55%	1951	1712	1687	1504
c6288	58	78	82	67	59	48
c7552	97.90%	93.55%	96.54%	97.30%	97.80%	97.37%

Each entry to the following tables gives the smallest number of test vectors required to achieve 100% coverage of all testable single stuck-at faults or the fault coverage obtained after applying 64K patterns to the corresponding CUT. The best results obtained for each benchmark circuit are shaded. Table 3 lists the best results obtained by each bit-serial test pattern generator assuming a 16-bit wide accumulator, adder or LFSR. In the case of the proposed scheme the given constant additive value $u = 40864$ and 3 other randomly selected additive constants were used for each benchmark circuit, while the number of test vectors that appear in the column referring to the LFSR, are taken from Table 1 of [11]. Data clearly show that the bit-sequences produced by the proposed scheme in most cases outperform both the LFSR and the other arithmetic module based test pattern generators. From Table 3 we can see that, in the case of 16-bit wide adder, the Multiply&Add scheme gives significantly worse results than the other schemes. This is due to the fact that the period obtained by this scheme is equal to $2^{16/2} = 256$, which is too short.

Table 3. Results achieved by 32-bit TPGs

CUT	LFSR	[11]		[10] MAC	Proposed	
		Mult& Add	Mult& Acc		Simple	Enhanced
c432	445	415	353	470	381	338
c499	576	532	431	708	493	422
c880	7412	3389	1118	4427	3400	3117
c1355	1413	1641	1602	1554	1515	1432
c1908	6571	4428	9280	5237	3861	4447
c2670	90.66%	86.9%	95.6%	91.10%	91.08%	92.33%
c3540	90.88%	9377	22399	14671	99.94%	8680
c5315	2632	1605	1739	2373	2261	1589
c6288	88	73	84	64	62	67
c7552	97.84%	98.0%	98.8%	98.20%	97.97%	97.78%

Table 4. Results achieved by 16-bit TPGs

CUT	LFSR	[11]		[10] MAC	Proposed	
		Mult& Add	Mult& Acc		Simple	Enhanced
s27	34	22	20	21	19	15
s208	97.02%	80.77%	1932	4123	96.10%	4646
s298	242	147	214	231	201	198
s344	116	122	120	98	90	106
s349	128	87	118	135	105	124
s382	236	99.87%	207	221	99.87%	306
s386	2478	85.88%	2768	2866	2122	2552
s420	88.01%	68.88%	86.46%	90.39%	81.88%	92.36%
s444	93.76%	99.31%	242	263	303	240
s510	99.31%	99.60%	1014	580	99.02%	598
s526	4328	89.49%	99.86%	9872	6243	8406
s641	98.74%	95.94%	98.51%	98.74%	98.59%	98.59%
s713	98.82%	93.94%	98.60%	98.81%	98.67%	98.89%
s820	93.78%	78.66%	98.98%	16388	83.78%	18671
s832	93.62%	78.32%	96.85%	13752	84.09%	18954
s838	62.47%	47.17%	58.95%	61.80%	58.90%	63.22%
s953	97.53%	81.22%	99.12%	99.94%	89.77%	99.95%
s1196	99.79%	80.73%	99.79%	99.71%	99.96%	99.87%
s1238	99.87%	76.96%	97.24%	99.54%	99.83%	99.91%
s1423	99.85%	94.80%	99.75%	99.86%	99.82%	16437
s1488	3513	91.06%	6343	3685	3595	3974
s1494	3729	91.45%	4673	3397	3671	4004

In Table 4, bit serial test pattern generators assuming a 32-bit wide accumulator, adder or LFSR are compared. With respect to the proposed scheme, since there is no

information about the proper selection of a constant additive value in order to achieve a maximal period, we randomly selected 8 constant inputs and tried 5 different initial states *s(0)* for each benchmark circuit in order to complete the needed 40 experiments. The results that appear in columns 3-4 were taken from [11].

The increase in the accumulator size has clearly resulted in certain improvements in the obtained fault coverage and in many cases has lead to a considerable reduction of the required test length to achieve 100% fault coverage, in comparison to the results obtained with a 16-bit wide accumulator. We can observe that although the selected constant additive value does not offer the maximal period, the proposed test pattern generator still offers an effective solution.

Table 5. Results achieved by 32-bit TPGs

CUT	LFSR	[11]		[10] MAC	Proposed	
		Mult& Add	Mult& Acc		Simple	Enhanced
s27	14	21	10	19	16	12
s208	3268	1670	1802	3818	1780	2324
s298	237	198	256	257	218	231
s344	213	115	94	157	95	92
s349	125	113	102	102	106	99
s382	268	208	243	320	264	263
s386	2224	1525	2312	1862	1706	1599
s420	94.56%	94.54%	97.82%	93.45%	94.87%	96.40%
s444	226	170	232	253	238	306
s510	413	504	418	481	571	514
s526	6438	5509	9631	8317	7254	9051
s641	99.53%	98.82%	98.90%	98,74%	98.67%	99.69%
s713	99.56%	98.89%	99.58%	99.63%	98.72%	99.61%
s820	11633	14083	14295	19855	14662	15442
s832	16533	14095	8746	9684	20105	15311
s838	68.08%	64.98%	71.59%	63.86%	65.72%	65.57%
s953	32987	34152	21804	27556	29217	19350
s1196	50722	44213	52182	60023	61832	33671
s1238	55124	54283	59874	63125	55481	58126
s1423	13124	14188	26433	22984	17887	16953
s1488	3760	4107	3691	4002	3751	3688
s1494	3920	3317	2998	4426	3998	3871

In the same way, Tables 5 and 6 present the best results obtained after performing 40 experiments on the ISCAS'89 benchmark circuits using the bit-serial test pattern generators with 16 and 32 bit wide accumulators, adders of LFSRs respectively. Once more we verify the effectiveness of the proposed scheme.

5. Conclusions

In this paper we have shown that an accumulator behaving as a NLFSR can be used efficiently for bit serial test pattern generation and in most cases, compares favorably to LFSR and other arithmetic function based bit serial sequence generators. Furthermore, our scheme has the advantage of wider applicability against the other arithmetic function based bit serial sequence generators proposed in [10, 11] since it can be applied even in circuits that an accumulator-multiplier or adder-multiplier configuration [10, 11] is not available. Finally, taking into account the suitability of the proposed scheme for parallel and serial test response compaction [12, 13] we conclude that the same scheme, depending on the test session, can be used effectively either as a test pattern generator or a test response compactor.

Acknowledgements

The authors would like to thank Dr. Y. Stamatiou for the useful discussion on randomness metrics.

References

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, NY, 1990.

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

[3] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing for Digital, Memory & Mixed Signal VLSI circuits,* Kluwer Academic Publishers, 2000.

[4] H. J. Wunderlich, *BIST for systems-on-a-chip*, Integration, The VLSI Journal, vol. 26, no.1-2, pp. 55-78, Dec. 1998.

[5] J.Rajski and J.Tyszer, *Arithmetic Built–In Self–Test for Embedded Systems*, Prentice Hall, 1998.

[6] Rajski J., Tyszer J., *Test Response Compaction in Accumulators with Rotate Carry Adders*, IEEE Trans. on CAD, vol. 12, no.4, pp. 531-539, April 1993.

[7] Stroele A. P., *Test Response Compaction Using Arithmetic Functions*, Proc. of IEEE VTS, pp. 380-386, 1996.

[8] Gupta S., Rajski J. and Tyszer J., *Arithmetic Additive Generators of Pseudo-Exhaustive Test Patterns*, IEEE Trans. on Comp., vol. 45, no. 8, pp. 939-949, Aug. 1996.

[9] Stroele A. P., *BIST Pattern Generators Using Addition and Subtraction Operations*, JETTA, vol. 11, pp. 69-80, Aug. 1997.

[10] J.Rajski and J.Tyszer, *Multiplicative Window Generators of Pseudo–random Test Vectors*, Proc. of European Design and Test Conference, pp. 42-48, 1996.

[11] A.P.Stroele, *Bit Serial Pattern Generation and Response Compaction Using Arithmetic Functions*, Proc. of 16th IEEE VLSI Test Symposium, pp 78–84, 1998.

[12] D. Bakalis, D. Nikolos and X. Kavousianos, *Test Response Compaction by an Accumulator behaving as a Multiple Input Non–Linear Feedback Shift Register*. Proc. of IEEE ITC, pp. 804-811, 2000.

[13] D. Bakalis, D. Nikolos, H. T. Vergos and X. Kavousianos, *On Accumulator-Based Bit-Serial Test Response Compaction Schemes*, Proc. of 2nd IEEE ISQED, pp. 350-355, 2001.

[14] D.E.Knuth, *The Art of Computer Programming*, Vol. 2, Addison-Wesley, 1981.

[15] R. Gold, *Optimal Binary Sequences for Spread Spectrum Multiplexing*, IEEE Trans. on Information Theory, vol. IT-B, pp. 619-621, October 1967.

[16] F. Brglez and H. Fujiwara, *A neutral netlist of 10 combinational benchmark circuits and a target translator in FORTRAN*, Proc. of IEEE ISCAS, 1985.

[17] F. Brglez, D.Bryan and K.Kozminski, *Combinational Profiles of Sequential Benchmark Circuits*, Proc. of IEEE ISCAS, pp. 1929–1934, 1989.