[5] L. C. N. de Vreede, H. C. de Graaff, J. A. Willemen, W. van Noort, R. Jos, L. E. Larson, J. W. Slotboom, and J. L. Tauritz, "Bipolar transistor epilayer design using the MAIDS mixed-level simulator," *IEEE J. Solid-State Circuits*, vol. 34, no. 9, pp. 1331–1338, Sep. 1999.
[6] T. Grasser, "Mixed-mode device simulation," Ph.D. dissertation, Inst. Microelectrons., Technischen Universität Wien, Vienna, Austria, May 1999. [Online]. Available: http://www.iue.tuwien.ac.at/phd/grasser/
[7] K. S. Kundert, "Introduction to RF simulation and its application," *IEEE J. Solid-State Circuits*, vol. 34, no. 9, pp. 1298–1319, Sep. 1999.
[8] Y. Hu and K. Mayaram, "Periodic steady-state analysis for coupled device and circuit simulation," in *Proc. SISPAD*, Sep. 2000, pp. 90–93.
[9] ——, "Harmonic balance analysis for coupled device and circuit simulation," in *Proc. RAWCON*, Aug. 2001, pp. 137–140.
[10] ——, "An efficient algorithm for large-signal frequency-domain coupled device and circuit simulation," in *Proc. ISCAS*, May 2002, vol. 5, pp. 329–332.
[11] ——, "A comparison of non-quasi-static and quasi-static harmonic balance implementations for coupled device and circuit simulation," in *Proc. CICC*, Sep. 2003, pp. 99–102.
[12] B. Troyanovsky, "Frequency domain algorithms for simulating large signal distortion in semiconductor devices," Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Stanford, CA, Nov. 1997.
[13] F. M. Rotella, G. Ma, Z. Yu, and R. W. Button, "Modeling, analysis, and design of RF LDMOS devices using harmonic-balance device simulation," *IEEE Trans. Microw. Theory Tech.*, vol. 48, no. 6, pp. 991–999, Jun. 2000.
[14] F. Filicori, G. Vannini, and V. A. Monaco, "A nonlinear integral model of electron device for HB circuit analysis," *IEEE Trans. Microw. Theory Tech.*, vol. 40, no. 7, pp. 1456–1465, Jul. 1992.
[15] D. Mirri, G. Iuculano, F. Filicori, G. Pasini, G. Vannini, and G. P. Gualtieri, "A modified Volterra series approach for nonlinear dynamic systems modeling," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 49, no. 8, pp. 1118–1128, Aug. 2002.
[16] Y. Hu and K. Mayaram, "A modified-Volterra-series technique for improving the accuracy of quasi-static harmonic balance analysis in coupled device and circuit simulation," in *Proc. CICC*, Oct. 2004, pp. 125–128.
[17] P. Feldmann, B. Melville, and D. Long, "Efficient frequency domain analysis of large nonlinear analog circuits," in *Proc. CICC*, May 1996, pp. 461–464.
[18] P. Rodrigues, *Computer-Aided Analysis of Nonlinear Microwave Circuits*. Norwood, MA: Artech, 1998.
[19] S. E. Laux, "Techniques for small-signal analysis of semiconductor devices," *IEEE Trans. Electron Devices*, vol. ED-32, no. 10, pp. 2028–2037, Oct. 1985.
[20] P. N. Ashar, "Implementation of algorithms for the periodic-steady-state analysis of nonlinear circuts," Univ. California, Berkeley, Memo. UCB/ERL M89/31, Mar. 1989.
[21] D. O. Pederson and K. Mayaram, *Analog Integrated Circuits for Communication*. Boston, MA: Kluwer, 1991.
[22] K. S. Kundert and A. Sangiovanni-Vincentelli, *Sparse 1.3: A Sparse Linear Equation Solver*. Berkeley: Univ. California, 1988.
[23] T. A. Davis, "UMFPACK Version 4.3 user guide," CIS Dept., Univ. Florida, Gainesville, Tech Rep. TR-03-008, 2003. [Online]. Available: http://www.cise.ufl.edu/research/sparse/umfpack/
[24] F. Bonani, S. D. Guerrieri, G. Ghione, and M. Pirola, "A TCAD approach to the physics-based modeling of frequency conversion and noise in semiconductor devices under large-signal forced operation," *IEEE Trans. Electron Devices*, vol. 48, no. 5, pp. 966–977, May 2001.
[25] J. E. Sanchez, G. Bosman, and M. E. Law, "Two-dimensional semiconductor device simulation of trap-assisted generation-recombination noise under periodic large-signal conditions and its use for developing cyclostationary circuit simulation models," *IEEE Trans. Electron Devices*, vol. 50, no. 5, pp. 1353–1362, May 2003.
[26] K. Mayaram, J. Chern, and P. Yang, "Algorithms for transient three-dimensional mixed-level circuit and device simulation," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 12, no. 11, pp. 1726–1733, Nov. 1993.

On Obtaining Maximum-Length Sequences for Accumulator-Based Serial TPG

D. Kagaris, P. Karpodinis, and D. Nikolos

***Abstract*—Arithmetic-function modules, which are available in many circuits, can be utilized to generate test patterns and compact test responses. An accumulator-based scheme along with a procedure to find maximum-length nonlinear sequences for bit-serial test-pattern generators is proposed. The proposed scheme achieves good fault coverage with low hardware overhead and short test sequences.**

***Index Terms*—Arithmetic BIST, built-in self-test (BIST), nonlinear test pattern generation (TPG).**

I. Introduction

Built-In Self-Test (BIST) [1]–[4] is a design for a testability strategy that reduces the need for an external test and can be used to determine faulty parts at all levels in the hierarchy (core to chip to board to system) during manufacturing and to diagnose problems in the field. In BIST test-pattern generation, response monitoring and evaluation are handled on chip with the use of extra hardware structures. Common BIST schemes used in practice 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]–[17] 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 environments have been considered [6]–[17]. In this paper, we concentrate on the arithmetic test-per-scan BIST environment. Test-per-scan BIST can find application in sequential circuits with scan paths, embedded cores with an isolation ring, circuits with boundary scan, and portions of multichip 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]. Although the fault coverage can be improved using a parallel/serial accumulator (in which case the register of the parallel accumulator is modified to function as a shift register), it remains generally low even for long test sequences. Therefore, these simple arithmetic units are not

Manuscript received December 16, 2004; revised April 23, 2005 and August 15, 2005. This work was supported by the European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II), and the Program PYTHAGORAS. This paper was presented in part in *Proceedings of the 10th IEEE International On-Line Testing Symposium*, Funchal, Portugal, July 2004, pp. 193–198. This paper was recommended by Associate Editor S. Hellebrand.

D. Kagaris is with the Electrical and Computer Engineering Department, Southern Illinois University, Carbondale, IL 62901 USA (e-mail: kagaris@engr.siu.edu).

P. Karpodinis is with the Computer Engineering and Informatics Department, University of Patras, Patras 26500, Greece (e-mail: karpodin@ceid.upatras.gr).

D. Nikolos is with the Computer Engineering and Informatics Department, University of Patras, Patras 26500, Greece, and also with the Computer Technology Institute, Patras 26221, Greece (e-mail: nikolosd@cti.gr).

Digital Object Identifier 10.1109/TCAD.2006.870860

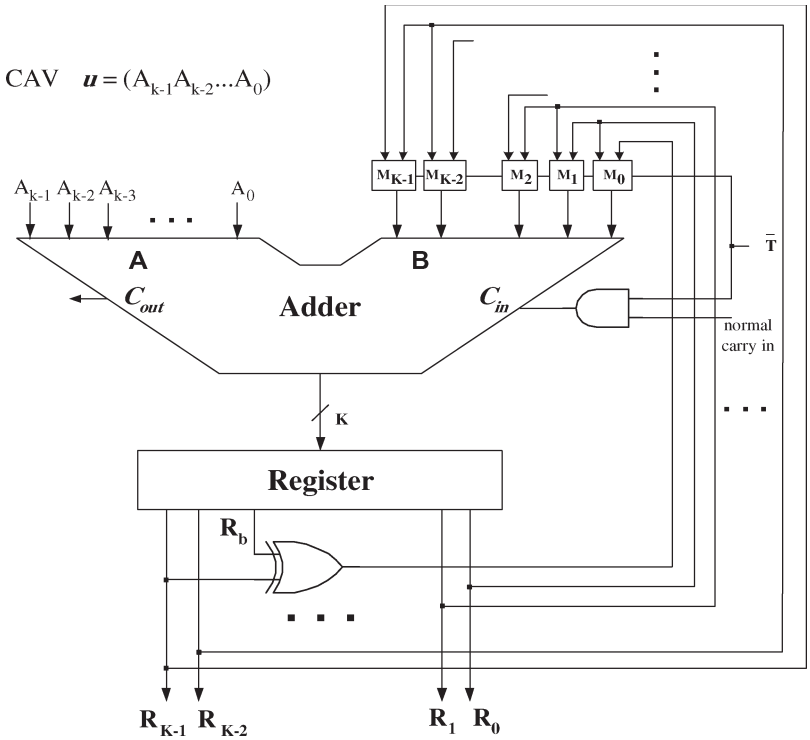

Fig. 1. Proposed ACC-XOR scheme.

efficient 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 proposed in [10] and [11]. These schemes, compared to an LFSR bit-serial test-pattern generator (TPG), have the advantage of achieving similar fault coverage with similar number of test patterns, while not requiring any hardware overhead, as they are already part of the functional circuit. The disadvantages of the schemes proposed in [10] and [11] can be summarized as follows: 1) their applicability is limited to applications in which the required configuration of the adder–multiplier or accumulator–multiplier is available and 2) since a multiplier–adder or multiplier–accumulator is used for test-pattern generation, these schemes have increased power and energy consumption during testing. A comparative study of the schemes proposed in [10] and [11] can be found in [15].

Recently, it was shown [12] that an accumulator can be modified to operate in test mode as a nonlinear feedback shift register (NLFSR) and that it can be used effectively for bit-serial test-pattern generation [15] as well as for test-response compaction [12], [18]. The length and quality of the test bit sequences generated by the above scheme depend on the suitable selection of a constant additive value (CAV) u. Several CAVs have been derived experimentally in [15] for accumulator size $K = 5$ up to $K = 20$ (note: the actual state bits in [15] are $K + 1$ as an extra flip-flop is used besides the K-bit accumulator register). However, finding a suitable pair of initial value and CAV such that the period of bit sequences is maximum requires exhaustive searching. Moreover, for some values of K (such as 7, 10, 12, 13, 17, and 20), there are no CAVs that ensure maximum period of bit sequences. Hereafter, the above scheme will be referred to as accumulator-based NLFSR (ANLFSR). A modification having better random properties was also presented in [15], where the most significant bits of the upper and the lower half of the register are XORed together to yield the test bit sequence. This scheme will be referred to as enhanced ANLFSR (EANLFSR).

In this paper, we propose a slightly different accumulator-based configuration that compares favorably to LFSRs and the other arithmetic-function-based bit-serial sequence generators with respect to the attainable fault coverage and the required test length.

The rest of the paper is organized as follows. The next section presents the new accumulator-based bit-serial test-pattern-generation scheme. Section III gives comparative experimental results of the proposed scheme with previous schemes, and Section IV concludes.

II. Proposed Scheme

A. Hardware Description

The proposed test-pattern-generator scheme is shown in Fig. 1. In normal mode, i.e., $T = 0$, the ith bit of the register, denoted by R_i, $0 \leq i \leq K-1$, drives the ith bit of the B input of the adder B_i. In test mode, bit R_i, $0 \leq i \leq K-2$, drives the B_{i+1} input of the adder. The inputs A_i, $0 \leq i \leq K-1$, of the adder are set to the CAV value u ($u \leq 2^K - 1$). The most significant bit of the register R_{K-1}, along with one other appropriately selected bit R_b, $0 \leq b \leq K-2$, drive the extra XOR gate. The output of the extra XOR gate drives the least significant bit B_0 of the adder.

Let $s(t)$ be a K-bit vector denoting the state of the K-bit register. Also, let X denote the output of the extra XOR gate. Then, the next state $s(t+1)$ can be described by

$$s(t+1) = [2s(t) + u + X] \mod 2^K. \tag{1}$$

(The above operation can optionally be implemented in a software in a microprocessor environment, as is the case also for the schemes in [10], [11], and [15].) We will refer to the accumulator-XOR (ACC-XOR) scheme with accumulator size K, bit position for the XOR b, and CAV u as $A(K, b, u)$. The interest is for $A(K, b, u)$ schemes with maximum period sequences. An ACC-XOR scheme with period $2^K - 1$ will be referred to as "primitive," borrowing the term from the linear case.

It can be observed that by keeping the CAV value to zero and by initializing the register to a nonzero value, the resulting scheme

[i.e., $A(K, b, 0)$] functions as a trinomial K-bit LFSR. We note that a separate LFSR could be embedded by inserting the multiplexers in the register. However, in this case, the delay of the accumulator, and most likely the delay of the whole circuit, would be affected. In the proposed scheme, the insertion of the multiplexers in the feedback path may not deteriorate the delay of the whole circuit, as the feedback path of the accumulator is less likely to be critical. Moreover, for $u \neq 0$, the use of nonlinearity in the generation of the test bit sequence gives better results with respect to test length and fault coverage, as experimental results show.

Stroele has proven [9] that if the transition diagram of an arbitrary finite-state machine with K-state bits contains state sequences of period $2^K - 1$, then the generated bit sequence of each bit position i, $0 \leq i \leq k-1$, has period $2^K - 1$. Therefore, when the proposed TPG, with K state bits, operates under a constant input and produces a state cycle of length $2^K - 1$, then the sequence generated from the jth bit position, $0 \leq j \leq K-1$, also has period $2^K - 1$. For bit-serial test-pattern generation, one can choose, in principle, any of these bit positions. We have chosen to use the most significant bit of the register (R_{K-1}) as the source of the random test sequence, based upon the observation in [19], which states that the least significant bits of a pseudorandom number sequence are much less "random" than the most significant ones.

The hardware overhead required for the proposed accumulator scheme is equal to $1.7K + 3.3$ gate equivalents. The hardware overhead for the ANLFSR and EALFSR schemes in [15] is equal to $1.7K + 1.3$ and $1.7K + 3.3$ gate equivalents, respectively, under the provision that the external D flip-flop and the XOR gate driving it have been incorporated in the accumulator logic. The hardware overhead of the multiply-and-accumulate (MAC) [10], multiply–add, and multiply–accumulate schemes [11] is zero. However, these schemes assume the existence of a multiplier–accumulator or multiplier–adder pair in the circuit. In the proposed scheme, only the existence of an accumulator is required. In most LFSR-based schemes, the LFSR is a dedicated circuit. In that case, the hardware required for the implementation is equal to $3.6K + 2m$, where m is the number of the XOR gates, as determined by the characteristic polynomial of the LFSR. It can be noted that the hardware overhead for modifying the accumulator in the proposed scheme is well less than half the cost of a separate LFSR. In the case that an existing register with size K is modified to function in test mode as a K-bit LFSR, the cost is equal to $1.7K + 2m$ equivalent gates. In the case of a trinomial LFSR, the latter cost becomes $1.7K + 2$, which is similar to the cost of our scheme.

We also note that in an accumulator-based scheme of size K, a number q, $q < K$, of least significant bits may prove enough for the generation of a test sequence with the desired characteristics. In that case, the hardware overhead of the proposed scheme can be reduced by setting, in test mode, $A_{K-1} = A_{K-2} = \cdots = A_{q-1} = 0$ (see Fig. 1), feeding the XOR gate with register bits R_{q-1} and R_b, $b < q-1$, and inserting multiplexers only at the q least significant inputs of adder input B.

B. Finding Primitive $A(k, b, u)$ Schemes

A brute-force approach to test for primitive $A(K, b, u)$ schemes is prohibitive, as it requires $K \cdot 2^k \cdot 2^k$ simulation steps. In the following, we propose a procedure to speed up the search.

Lemma 1: The result of XORing the bit at position b (starting from zero) with the most significant bit (at position $K-1$) is given by

$$r = \left(\left\lfloor \frac{s}{2^{K-1}} \right\rfloor + \left\lfloor \frac{s}{2^b} \right\rfloor \right) \bmod 2.$$

Proof: The result of XORing the bit at position b (indexing starting from 0) with the most significant bit (at position $K-1$) can, by definition, be expressed mathematically as

$$r = \left(\left\lfloor \frac{s}{2^{K-1}} \right\rfloor + \left\lfloor \frac{s \bmod 2^{b+1}}{2^b} \right\rfloor \right) \bmod 2. \tag{2}$$

Let the quotient and remainder of s in the division by 2^{b+1} be q_1 and r_1, respectively; that is, $s = q_1 \cdot 2^{b+1} + r_1$. Also, let q_2 and r_2 be the quotient and remainder of r_1 in the division by 2^b; that is, $r_1 = q_2 \cdot 2^b + r_2$. Finally, let q_3 and r_3 be the quotient and remainder of s in the division by 2^b; that is, $s = q_3 \cdot 2^b + r_3$. Then, $s = q_1 \cdot 2^{b+1} + q_2 \cdot 2^b + r_2 = q_3 \cdot 2^b + r_3$, from which it follows simultaneously that $r_2 = r_3$ and $q_3 = 2q_1 + q_2$ due to the integer-division property. In particular, $q_3 \bmod 2 = q_2 \bmod 2$, and so by replacing q_3 and q_2 with their equivalent expressions, we have $\lfloor s \bmod 2^{b+1}/2^b \rfloor \bmod 2 = \lfloor s/2^b \rfloor \bmod 2$. Therefore, (2) can be rewritten as $r = (\lfloor s/2^{K-1} \rfloor + \lfloor s/2^b \rfloor) \bmod 2$. ■

The search for ACC-XOR schemes of a given K resulting in maximum-length sequences involves, in principle, the trial of all pairs of b and u values, where $0 \leq b \leq K-2$, $1 \leq u \leq 2^K - 1$, and the identification through simulation of the cyclic structure of the resulting states. To speed up the search, we eliminate several $A(K, b, u)$ schemes based on the following criteria:

1) elimination of certain $A(K, b, u)$ schemes that cannot be primitive, based only on mathematical formulas, and not on simulation;
2) elimination of certain $A(K, b, u')$ schemes that cannot be primitive, based on the identification, through simulation of a certain nonprimitive $A(K, b, u)$ scheme with $u' > u$.

1) Elimination Without Simulation: The nonprimitiveness of several $A(K, b, u)$ schemes can be determined *a priori* through mathematical formulas, based on the following lemmas and theorems.

Lemma 2: If $\lfloor (2^K - u)/2^b \rfloor$ is odd, then $A(K, b, u)$, with $0 \leq b < K-1$ and $0 < u < 2^{K-1}$, contains a cycle of length one (consisting of state $s = 2^K - u$).

Proof: Let s be a state such that $s = 2^K - u$. Then, the next state is $s' = [2s + u + (\lfloor s/2^{K-1} \rfloor + \lfloor s/2^b \rfloor) \bmod 2] \bmod 2^K$; that is, $s' = [2 \cdot 2^K - 2 \cdot u + u + (\lfloor (2^K - u)/2^{K-1} \rfloor + \lfloor (2^K - u)/2^b \rfloor) \bmod 2] \bmod 2^K$. Since $0 < u < 2^{K-1}$, we have that $\lfloor (2^K - u)/2^{K-1} \rfloor = 1$. Since $\lfloor (2^K - u)/2^b \rfloor$ is odd, the term inside the parentheses above is even, and so it is equal to $0 \bmod 2$. Therefore, $s' = [2 \cdot 2^K - 2 \cdot u + u] \bmod 2^K = (2^K + (2^K - u)) \bmod 2^K = 2^K - u = s$. ■

Similarly, the following lemma can be shown.

Lemma 3: If $\lfloor (2^K - u - 1)/2^b \rfloor$ is even, then $A(K, b, u)$, with $0 \leq b < K-1$ and $u < 2^{K-1}$, contains a cycle of length one (consisting of state $2^K - u - 1$).

If the relations in the Lemmas 2 and 3 above hold simultaneously for some $A(K, b, u)$, then we have the following theorem.

Theorem 1: If for some $A(K, b, u)$ scheme, with $0 \leq b < K-1$ and $0 < u < 2^{K-1}$, $\lfloor (2^K - u)/2^b \rfloor$ is odd and $\lfloor (2^K - u - 1)/2^b \rfloor$ is even, then that scheme is not primitive.

A similar theorem can also be shown as follows.

Theorem 2: If for some $A(K, b, u)$ scheme, with $0 \leq b < K-1$ and $2^{K-1} < u < 2^K$, $\lfloor (2^K - u)/2^b \rfloor$ is even and $\lfloor (2^K - u - 1)/2^b \rfloor$ is odd, then that scheme is not primitive.

Notice that Theorem 1 refers to CAVs with the most significant bit equal to zero, while Theorem 2 refers to CAVs with the most significant bit equal to one. As an illustration, we have that scheme $A(6, 3, 24)$ is not primitive by Theorem 1 as it contains two cycles of length one on states 40 and 39. Also, scheme $A(6, 3, 48)$ is not primitive according to Theorem 2, as it contains two cycles of length 1 on states 16 and 15.

TABLE I
SAMPLE OF BIT POSITIONS AND CAVS THAT ENSURE MAXIMUM PERIOD FOR EACH K

K	(Bit position, CAV)
6	(0, 32) (2, 6) (2, 26) (2, 34) (2, 62) (4, 32) (4, 36) (4, 60)
7	(0, 64) (2, 26) (2, 38) (3, 6) (3, 126) (4, 24) (4, 40) (5, 66)
8	(2,190) (2, 194) (3, 54) (3, 74) (3, 180) (3, 204) (4, 170) (4, 214)
9	(4, 46) (4, 444) (6, 8) (6, 80) (6, 228) (6, 248) (6, 506) (7, 472)
10	(2, 118) (2, 1022) (4, 202) (5, 656) (6, 16) (6, 512) (7, 976) (8, 960)
11	(2, 234) (2, 1834) (3, 1742) (4, 2036) (5, 1914) (7, 310) (7, 1760) (8, 1024)
12	(4, 890) (4, 1158) (6, 918) (6, 3362) (8, 2706) (9, 824) (9, 3772) (10, 3838)
13	(3, 2236) (5, 2588) (6, 3558) (7, 2248) (8, 3614) (9, 3104) (10, 2318) (10, 7904)
14	(2, 4198) (3, 4508) (6, 6738) (7, 4284) (8, 7954) (9, 5712) (10, 6834) (11, 7582)
15	(2, 8830) (3, 9926) (5, 15230) (6, 12266) (9, 13896) (11, 11558) (12, 16092) (13, 31310)
16	(4, 11260) (6, 27448) (7, 23210) (8, 29856) (11, 13852) (12, 58458) (13, 60296) (14, 57702)
17	(4, 40802) (5, 20124) (6, 10822) (8, 30796) (10, 33194) (11, 36104) (12, 49770) (13, 51286)
18	(2, 63158) (3, 76012) (4, 88108) (5, 87180) (6, 94278) (7, 58836) (8, 88344) (9, 95318)
19	(3, 163254) (5, 26166) (6, 49538) (8, 33312) (9, 44688) (11, 22674) (12, 50678) (13, 39898) (14, 256484)
20	(2, 267854) (3, 262140) (4, 337062) (6, 367924) (7, 272004) (8, 335756) (9, 437864) (10, 446656)
21	(2, 529762) (3, 213338) (4, 400354) (7, 195074) (8, 181808) (8, 866768) (9, 283654) (10, 174322)
22	(2, 1652154) (3, 1628926) (6, 1089634) (7, 1652382) (11, 1441998) (13, 1023958) (14, 1522736) (15, 1738414)
23	(4, 2097144) (5, 2085914) (6, 736714) (7, 1986620) (9, 1038914) (10, 2094588) (13, 2367380) (14, 2794960)
24	(2, 4610110) (3, 6261634) (6, 6439946) (7, 6455686) (9, 6200834) (10, 5898892) (12, 5882022) (13, 6476080)
25	(3, 8985292) (5, 10050842) (6, 12105358) (9, 13283262) (10, 9474304) (11, 13327194) (12, 6907358)
26	(2, 20739558) (4, 14241020) (6, 20085030) (7, 15244154) (10, 7919528)
27	(2, 33554430) (8, 51181336)
28	(4, 112775046) (6, 100350538) (7, 122824516) (8, 17357136)
29	(2, 133652770) (5, 153470694)
30	(2, 516563642) (4, 254504228) (6, 14501472)
31	(3, 533119780) (4, 294390828)
32	(7, 1197375798) (7, 1338246558) (7, 1430906754)

2) Elimination Through Simulation: The general idea here is the following. Assume that for some scheme $A(K, b, u')$, we have found, by simulation, that it contains a cycle of length $l < 2^K - 1$ and the states in the cycle satisfy certain conditions specified below. Then, a cycle of the same length will also be present in scheme $A(K, b, u' + x)$, for an appropriate value of x, preventing $A(K, b, u' + x)$ from being primitive. This is based on the following lemma.

Lemma 4: Successive states $s_1, s_2, \ldots, s_l$ of $A(K, b, u)$ correspond to successive states $s'_1, s'_2, \ldots, s'_l$ of $A(K, b, u')$, with $s'_i = s_i - x$, where $u' - u = x > 0$, provided that

$$\left(\left\lfloor \frac{s_i}{2^{K-1}} \right\rfloor + \left\lfloor \frac{s_i}{2^b} \right\rfloor\right) \bmod 2 = \left(\left\lfloor \frac{s_i - x}{2^{K-1}} \right\rfloor + \left\lfloor \frac{s_i - x}{2^b} \right\rfloor\right) \bmod 2.$$

Proof: Consider states s_1 and s'_1 of $A(K, b, u)$ and $A(K, b, u') = A(K, b, u + x)$, respectively, such that $s'_1 = s_1 - x$. The next state s'_{i+1} is given as

$$s'_{i+1} = \left(2 \cdot s'_i + u' + \left(\left\lfloor \frac{s'_i}{2^{K-1}} \right\rfloor + \left\lfloor \frac{s'_i}{2^b} \right\rfloor\right) \bmod 2\right) \bmod 2^K.$$

Since $s'_i = s_i - x$ and $u' - u = x$, we get

$$s'_{i+1} = \Bigg(2 \cdot (s_i - x) + (u + x) + \left(\left\lfloor \frac{s_i - x}{2^{K-1}} \right\rfloor + \left\lfloor \frac{s_i - x}{2^b} \right\rfloor\right) \bmod 2\Bigg) \bmod 2^K.$$

However

$$s_{i+1} = \left(2 \cdot s_i + u + \left(\left\lfloor \frac{s_i}{2^{K-1}} \right\rfloor + \left\lfloor \frac{s_i}{2^b} \right\rfloor\right) \bmod 2\right) \bmod 2^K$$

TABLE II
FAULT COVERAGE FOR $K = 17$ UNDER 64 000 VECTORS AND NUMBER OF VECTORS IN CASE OF A TIE

CUT	LFSR	[11]		[10] MAC	[13]		Proposed
		Mul&Add	Mul & Acc		ANLFSR	EANLFSR	
s27	100% 14	100% 19	100% 21	100% 25	100% 22	100% 17	100% 14
s208	100% 2457	91.74%	99.08%	100% 3154	96.33%	100% 1745	100% 1591
s298	100% 229	99.83%	100% 374	100% 559	100% 292	100% 174	100% 162
s349	99.41% 109	99.41% 101	99.41% 129	99.41% 201	99.41% 133	99.41% 74	99.41% 101
s382	100% 318	100% 211	100% 385	100% 501	99.87	100% 323	100% 296
s386	100% 1876	88.21%	100% 2517	100% 2832	100% 2126	100% 1811	100% 2263
s420	94.32%	69.76%	81.22%	90.39%	82.10%	93.23%	94.65%
s444	97.52% 222	97.52% 202	97.52% 262	97.52% 287	97.52% 255	97.52% 267	97.52% 353
s510	100% 450	98.33%	100% 523	100% 917	99.41%	100% 558	100% 366
s526	99.91% 10680	88.02%	99.81%	99.91% 22461	99.91% 11182	99.91% 7894	99.91% 6029
s641	98.59%	95.64%	98.51%	98.82%	98.59%	98.82%	99.53%
s713	93.62%	90.74%	93.55%	93.83%	93.62%	93.83%	94.46%
s832	98.32%	78.07%	95.55%	98.98% 17047	90.93%	98.98% 17226	98.98% 15039
s838	64.98%	51.87%	57.14%	61.03%	58.64%	63.53%	65.09%
s953	99.84%	77.91%	93.18%	99.95%	89.77%	99.95%	24962
s1196	100% 58674	83.74%	95.40%	99.71%	100% 55131	99.88%	99.96%
s1238	96.77%	80.41%	92.29%	96.57%	96.77%	96.65%	96.73%
s1423	99.09% 23974	94.73%	98.38%	98.98%	99.05%	99.09% 24285	99.09% 13944
s1494	99.47% 3563	88.82%	99.13%	99.47% 4421	99.47% 3360	99.47% 3588	99.47% 3822
s5378	98.76%	88.21%	97.52%	98.38%	98.76%	98.84%	98.85%
s9234	88.32%	65.02%	85.59%	87.57%	87.29%	89.53%	89.43%
s13207	98.43%	78.39%	92.69%	90.57%	98.54%	98.60%	98.69%

from which it follows that

$$s'_{i+1} = s_{i+1} - x$$

provided that

$$\left(\left\lfloor \frac{s_i}{2^{K-1}} \right\rfloor + \left\lfloor \frac{s_i}{2^b} \right\rfloor\right) \bmod 2 = \left(\left\lfloor \frac{s_i - x}{2^{K-1}} \right\rfloor + \left\lfloor \frac{s_i - x}{2^b} \right\rfloor\right) \bmod 2.$$

■

Theorem 3: If scheme $A(K, b, u)$ contains a cycle $s_1, s_2, \ldots, s_l, s_1$ of length $l < 2^K - 1$ such that the cycle contains no state s_i, $0 < s_i < 2^{K-1}$, that is a multiple of 2^b, then a cycle of the same length containing states $s_1 - 1, s_2 - 1, \ldots, s_l - 1, s_1 - 1$ exists also in scheme $A(K, b, u + 1)$.

Proof: According to Lemma 4, the following relation must be satisfied (setting $x = 1$):

$$\left(\left\lfloor \frac{s_i}{2^{K-1}} \right\rfloor + \left\lfloor \frac{s_i}{2^b} \right\rfloor\right) \bmod 2 = \left(\left\lfloor \frac{s_i - 1}{2^{K-1}} \right\rfloor + \left\lfloor \frac{s_i - 1}{2^b} \right\rfloor\right) \bmod 2.$$

Therefore $>$

$$\left(\left\lfloor \frac{s_i}{2^{K-1}} \right\rfloor - \left\lfloor \frac{s_i - 1}{2^{K-1}} \right\rfloor + \left\lfloor \frac{s_i}{2^b} \right\rfloor - \left\lfloor \frac{s_i - 1}{2^b} \right\rfloor\right) \bmod 2 = 0.$$

The range of values for s_i is $[0, \ldots, 2^K - 1]$. Consider the following cases.

1) $s_i = 2^{K-1}$: The first term $\lfloor s_i/2^{K-1} \rfloor$ in the relation above equals one, the second term $\lfloor (s_i - 1)/2^{K-1} \rfloor$ equals zero, the third term $\lfloor s_i/2^b \rfloor$ is even, and the fourth term $\lfloor (s_i - 1)/2^b \rfloor$ is odd, making the overall sum even, and thus satisfying the condition.
2) $s_i = 0$: The first and third terms are zero. The second term equals $\lfloor -(1/2^{K-1}) \rfloor$ in this case, and since $\lfloor -a \rfloor = -\lceil a \rceil$ for any real number a, $\lfloor -(1/2^{K-1}) \rfloor = -\lceil 1/2^{K-1} \rceil = -1$. For the same reason, the fourth term equals one, making the overall sum even, and thus satisfying the condition.
3) $s_i \neq \lambda 2^b$, for any positive integer λ, i.e., s_i is not a multiple of 2^b. Then, the first and second terms are equal, and the third and fourth terms are equal also, making the overall sum even, and thus satisfying the condition.
4) $s_i = \lambda 2^b$, for any positive integer λ, $0 < \lambda < 2^{K-1-b}$, i.e., s_i is a multiple of 2^b other than zero or 2^{K-1}. Then, the first and second terms equal zero, the third term equals λ, and the fourth term equals $\lambda - 1$, making the overall sum odd, and thus violating the condition. ■

Theorem 4: If scheme $A(K, b, u)$ contains a cycle $s_1, s_2, \ldots, s_l, s_1$ of length $l < 2^K - 1$ such that the cycle contains no state s_i with $2^{K-1} \leq s_i < 2^{K-1} + 2^{b+1}$ or $0 \leq s_i < 2^{b+1}$, then a cycle

TABLE III
FAULT COVERAGE FOR $K = 25$ UNDER 64 000 VECTORS AND NUMBER OF VECTORS IN CASE OF A TIE

CUT	LFSR	[11]		[10] MAC	[13]		Proposed
		Mul&Add	Mul&Acc		ANLFSR	EANLFSR	
s27	100% 15	100% 13	100% 13	100% 25	100% 14	100% 16	100% 16
s208	100% 1010	100% 1946	100% 16140	100% 3935	100% 2269	100% 1249	100% 1266
s298	100% 219	100% 212	100% 197	100% 177	100% 273	100% 303	100% 174
s349	99.41% 126	99.41% 101	99.41% 80	99.41% 125	99.41% 174	99.41% 148	99.41% 100
s382	100% 255	100% 267	100% 358	100% 303	100% 247	100% 266	100% 160
s386	100% 2373	100% 1590	100% 2365	100% 3193	100% 2135	100% 1329	100% 1427
s420	95.85%	83.62%	84.17%	94.32%	80.23%	94.32%	97.27%
s444	97.52% 213	97.52% 203	97.52% 217	97.52% 222	97.52% 218	97.52% 261	97.52% 296
s510	100% 603	100% 573	100% 741	100% 768	100% 617	100% 462	100% 482
s526	99.91% 10099	99.14%	99.91%	99.91% 12101	99.91% 12280	99.91% 10906	99.91% 8245
s641	99.60%	98.43%	98.67%	98.74%	98.82%	99.53%	99.77%
s713	94.53%	93.48%	93.69%	93.76%	93.83%	94.46%	94.67%
s832	98.98% 11211	95.97%	98.98% 17961	98.98% 17842	98.98% 16413	98.98% 12851	98.98% 12642
s838	65.83%	57.84%	58.80%	62.79%	63.17%	65.25%	67.32%
s953	100% 19416	96.07%	100% 36859	100% 14599	100% 20503	100% 25757	100% 17828
s1196	99.96%	97.70%	99.21%	99.96%	43439	20590	37040
s1238	96.73%	94.55%	96.04%	96.73%	96.77% 43439	96.77% 20590	96.77% 36520
s1423	99.09% 16114	98.60%	99.09% 23222	99.05%	98.88%	99.09% 37178	99.09% 12766
s1494	99.47% 2422	99.43%	99.47% 6955	99.47% 4910	99.47% 4195	99.47% 3047	99.47% 4340
s5378	98.86% 54709	96.99%	98.81%	98.86% 63918	98.82%	98.86% 52860	98.86% 50969
s9234	89.71%	81.21%	88.31%	89.34%	89.10%	90.17%	89.60%
s13207	98.76%	88.98%	98.36%	98.60%	98.66%	98.71%	98.78%

TABLE IV
ADDITIONAL RESULTS FOR LARGER CIRCUITS: FAULT COVERAGE UNDER 64 000 VECTORS AND NUMBER OF VECTORS IN CASE OF A TIE

CUT	K=17			K=25			K=32		
	LFSR	EANLFSR	Proposed	LFSR	EANLFSR	Proposed	LFSR	EANLFSR	Proposed
s15850	94.60%	94.64%	94.83%	94.90%	94.96%	95.18%	94.72%	N/A	94.88%
s35932	89.69% 181	89.69% 187	89.69% 171	89.69% 149	89.69% 167	89.69% 171	89.69% 186	N/A	89.69% 164
s38417	97.29%	97.34%	97.20%	97.54%	97.37%	97.20%	97.41%	N/A	97.44%
s38584	95.24%	95.19%	95.31%	95.29%	95.23%	95.30%	95.24%	N/A	95.23%

of the same length containing states $s_1 - 2^{b+1}, s_2 - 2^{b+1}, \ldots, s_l - 2^{b+1}, s_1 - 2^{b+1}$ exists also in scheme $A(K, b, u + 2^{b+1})$.

Proof: According to Lemma 4, the following relation must be satisfied (setting $x = 2^{b+1}$):

$$\left(\left\lfloor \frac{s_i}{2^{K-1}} \right\rfloor + \left\lfloor \frac{s_i}{2^b} \right\rfloor\right) \bmod 2$$

$$= \left(\left\lfloor \frac{s_i - 2^{b+1}}{2^{K-1}} \right\rfloor + \left\lfloor \frac{s_i - 2^{b+1}}{2^b} \right\rfloor\right) \bmod 2.$$

Therefore

$$\left(\left\lfloor \frac{s_i}{2^{K-1}} \right\rfloor - \left\lfloor \frac{s_i - 2^{b+1}}{2^{K-1}} \right\rfloor + \left\lfloor \frac{s_i}{2^b} \right\rfloor - \left\lfloor \frac{s_i - 2^{b+1}}{2^b} \right\rfloor\right) \bmod 2 = 0.$$

Consider the following cases.

1) $2^{K-1} \le s_i < 2^{K-1} + 2^{b+1}$: Since the maximum value of s_i is $2^K - 1$, term $\lfloor s_i/2^{K-1} \rfloor$ in the relation above equals one, and term $\lfloor (s_i - 2^{b+1})/2^{K-1} \rfloor$ equals zero. Term $\lfloor (s_i - 2^{b+1})/2^b \rfloor$ equals $\lfloor (s_i/2^b) - 2 \rfloor = \lfloor s_i/2^b \rfloor - 2$. Therefore, the overall sum

TABLE V
STANDARD DEVIATION OF FAULT COVERAGE ($K = 17$)

CUT	Metrics	LFSR	[11]		[10]	[13]		Proposed
			Mul&Add	Mul&Acc	MAC	ANLFSR	EANLFSR	
s526	Max	99.91%	88.02%	99.81%	99.90%	99.90%	99.90%	99.91%
	Avg	99.71%	86.74%	99.52%	99.90%	99.72%	99.71%	99.91%
	Dev	0.14	1.32	0.30	0	0.59	0.52	0
s641	Max	98.59%	95.37%	98.51%	98.82%	98.59%	98.82%	99.53%
	Avg	97.96%	94.33%	98.35%	98.74%	98.42%	98.58%	98.87%
	Dev	0.64	0.72	0.17	0.11	0.48	0.23	0.34
s713	Max	93.62%	90.74%	93.55%	93.83%	93.62%	93.83%	94.46%
	Avg	93.06%	89.81%	93.41%	93.76%	93.47%	93.61%	93.87%
	Dev	0.58	0.65	0.15	0.10	0.43	0.21	0.30
s832	Max	98.32%	78.066%	95.55%	98.98%	90.93%	98.98%	98.98%
	Avg	93.04%	76.74%	91.38%	98.98%	85.02%	98.39%	98.98%
	Dev	6.82	0.95	4.40	0	3.21	2.13	0
s838	Max	64.98%	51.87%	57.14%	61.03%	58.63%	63.54%	65.09%
	Avg	60.28%	47.88%	50.13%	60.13%	56.63%	61.04%	62.34%
	Dev	3.41	2.54	7.34	1.28	2.17	3.08	1.27
s953	Max	99.84%	77.91%	93.18%	99.95%	89.768%	99.95%	100.00%
	Avg	94.99%	76.73%	86.62%	99.79%	88.75%	98.77%	99.45%
	Dev	4.81	1.42	6.91	0.22	1.59	2.34	0.58
s1196	Max	100.00%	83.74%	95.40%	99.71%	100.00%	99.88%	99.96%
	Avg	99.05%	79.59%	94.59%	99.60%	99.45%	99.10%	99.87%
	Dev	0.73	2.79	0.86	0.15	1.17	2.04	0.08
s1238	Max	96.77%	80.41%	92.29%	96.57%	96.77%	96.65%	96.73%
	Avg	95.90%	76.45%	91.50%	96.45%	96.23%	95.93%	96.66%
	Dev	0.70	2.50	0.83	0.17	1.20	1.93	0.05
s1423	Max	99.09%	94.73%	98.38%	98.98%	99.05%	99.09%	99.09%
	Avg	99.04%	93.65%	98.03%	98.96%	98.82%	98.93%	99.08%
	Dev	0.04	0.70	0.37	0.02	0.48	0.24	0.03
s1494	Max	99.47%	88.82%	99.13%	99.47%	99.47%	99.47%	99.47%
	Avg	99.47%	88.18%	97.19%	99.47%	99.42%	99.19%	99.47%
	Dev	0	0.48	2.05	0	0.16	0.29	0
s5378	Max	98.76%	88.21%	97.52%	98.38%	98.76%	98.84%	98.85%
	Avg	98.61%	88.01%	96.18%	98.36%	96.20%	96.30%	98.76%
	Dev	0.17	0.18	1.37	0.03	7.59	7.53	0.11
s9234	Max	88.32%	65.02%	85.59%	87.57%	87.29%	89.53%	89.43%
	Avg	86.16%	64.04%	79.13%	87.49%	81.90%	85.98%	89.05%
	Dev	1.45	0.91	6.62	0.11	8.82	9.69	0.19
s13207	Max	98.43%	78.39%	92.69%	90.57%	98.54%	98.60%	98.69%
	Avg	98.11%	76.88%	88.18%	90.18%	95.36%	95.92%	98.50%
	Dev	0.19	1.41	4.61	0.54	7.13	7.03	0.17

is odd; that is, these values of s_i violate the condition (for the special case $b = K - 2$, the value $s_i = 2^{K-1}$ also violates the condition as the term $\lfloor (s_i - 2^{b+1})/2^{K-1} \rfloor$ equals two).

2) $0 \leq s_i < 2^{b+1}$: Term $\lfloor s_i/2^{K-1} \rfloor$ equals zero (assuming $b < K - 2$). Term $\lfloor (s_i - 2^{b+1})/2^{K-1} \rfloor$ equals $\lfloor (s_i - 2^{b+1})/2^{K-1} \rfloor = \lfloor (2^{b+1} - s_i)/2^{K-1} \rfloor = 1$. The last two terms have even sum for the same reasons as in 1) above. Therefore, the overall sum is odd; that is, these values of s_i violate the condition (for the special case $b = K - 2$, the value $s_i = 2^{b+1}$ also violates the condition as the first term equals $\lfloor 2^{K-1}/2^{K-1} \rfloor = 1$, but the second term equals $\lfloor (2^{K-1} - 2^{K-1})/2^{K-1} \rfloor = 0$). For the remaining values of s_i, the first term equals the second term and the third term equals the fourth, making the overall sum even, and thus satisfying the condition. ■

The value of x in Theorem 2 is $x = 1$ and in Theorem 3 is $x = 2^{b+1}$. The eliminated $A(K, b, u')$ schemes are obtained as follows.

1) Assume that we have found, by simulation, that scheme $A(K, b, u)$ contains a cycle satisfying the condition in Theorem 3. Let d be the minimum amount among all states in the cycle by which a state can be decreased without becoming a multiple of 2^b greater than zero and smaller than 2^{K-1}. Then, we know that all schemes $A(K, b, u + 1), A(K, b, u + 2), \ldots, A(K, b, u + (d + 1))$ are also nonprimitive. For example, by examining (through simulation) scheme $A(7, 3, 30)$, we find that it contains a cycle [12, 55, 12] of length two. The maximum decrease in the values of the states in this cycle so that no state becomes a multiple of $2^3 = 8$ (other than zero or 64) is $d = 3$. Therefore, a cycle of the same length is also present in schemes $A(7, 3, 31), A(7, 3, 32), A(7, 3, 33)$ and $A(7, 3, 34)$, i.e., these schemes do not need to be examined as they are nonprimitive.

2) Assume that we have found, by simulation, that scheme $A(K, b, u)$ contains a cycle satisfying the condition in Theorem 4. Let d be the minimum number of times among all states in the cycle by which a state can be decreased by 2^{b+1} each time, without falling in the forbidden regions $[2^{K-1}, 2^{K-1} + 2^{b+1})$ or $[0, 2^{b+1})$. Then, we know that all schemes $A(K, b, u + 2^{b+1}), A(K, b, u + 2 \cdot 2^{b+1}), \ldots, A(K, b, u + (d + 1) \cdot 2^{b+1})$ are also nonprimitive. For example, by examining (through simulation) scheme $A(7, 3, 36)$, we find that it contains a cycle [32, 100, 109, 126, 32] of length four. The maximum multiple of $2^{3+1} = 16$ by which the values of the states in this cycle can be decreased so that no state falls in the forbidden regions [64, 80) or [0, 16) is 16, i.e., $d = 1$. Therefore, a cycle of the same length is also present in schemes $A(7, 3, 52)$ and $A(7, 3, 68)$, i.e., these schemes do not need to be examined as they are nonprimitive.

C. Representative Values

The search procedure using the above criteria is quite effective. For instance, it finds the two primitive schemes $A(21, 8, 181808)$ and $A(21, 8, 866768)$, which are the only possible ones for $K = 21$ and $b = 8$, by doing only 10 838 simulations instead of the $2^{21} - 1 = 2\,097\,151$ simulations (that is, one for each value of u, $1 \leq u \leq 2^K - 1$), which would be required under the brute-force approach. Note that each simulation in either case requires $2^K = 2^{21}$ basic steps to determine the cyclic structure of the candidate scheme $A(21, 8, u)$. Due to this, the search is still CPU extensive, but building a database of $A(K, b, u)$ schemes is a one-time cost.

In Table I, we present a sample of bit positions and CAVs that ensure maximum period for each K. As can be observed, the proposed scheme yields maximum-length sequences for all values of K from $K = 6$ up to $K = 32$ (in each case, we report at most eight representatives). In contrast, in the case of an ANLFSR or EANLFSR, maximum-length sequences do not exist for several accumulator sizes.

We finally note that in contrast with an LFSR where the looping state is always zero, the looping state in a primitive $A(K, b, u)$ scheme depends on the scheme and is determined by the following lemma.

Lemma 5: In any primitive $A(K, b, u)$ scheme, the looping state is either $s = 2^K - u$ or $2^K - u - 1$ (the actual state can be identified by a simulation of one cycle).

Proof: Since the scheme $A(K, b, u)$ is primitive, it contains a cycle of length $2^K - 1$ and a cycle of length one. Let s be the state in the latter cycle (looping state). From (1), we have that $s = [2s + u + X] \bmod 2^K$. Since $s < 2^K$, we get $0 = [s + u + X] \bmod 2^K$, from which $s = 2^K - u - X$, where X is the result of the XOR operation, which can be either one or zero. That is, the looping state is either $s = 2^K - u$ or $2^K - u - 1$, and can be identified by simulation of one cycle. ■

For example, in the primitive scheme $A(7, 2, 26)$, the looping state is $s = 128 - 26 = 102$, whereas in the primitive scheme $A(7, 2, 38)$, the looping state is $s = 128 - 38 - 1 = 89$.

III. Evaluation and Comparisons

In order to evaluate the quality of the test sequences generated by the proposed scheme, a variety of experiments were conducted using the ISCAS'89 benchmarks [20]. We assume that the primary inputs, the internal flip-flops, and the primary outputs 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 TPG over the total number of faults.

For each one of the bit-serial TPGs that are evaluated, 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 maximum number of different patterns can be applied to the circuit under test (CUT).

Two sets of experiments were performed. Using the 16- and 24-bit accumulator-based schemes in [15] as base cases, we compare their performance against 17- and 25-bit ACC-XOR schemes and 17- and 25-bit primitive trinomial LFSRs. The difference in the sizes by one is due to the fact that the schemes in [15] have an extra flip-flop that extends the size of the accumulator register. For each type of TPG scheme, 20 experiments with different configurations (in terms, accordingly, of primitive polynomial, input constants, CAV, etc.) and seeds were tried for each benchmark circuit. 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.

TABLE VI
SAMPLE OF ABSOLUTE DEVIATIONS OF π APPROXIMATION BY CESARO METHOD

NumWidth	LFSR	ANLFSR	EANLFSR	Proposed
8	0.008073	0.007657	0.007467	0.007701
9	0.001168	0.001767	0.001567	0.001335
10	0.002120	0.001056	0.001874	0.001883
11	0.000645	0.000192	0.000517	0.000156
12	0.000403	0.000039	0.000961	0.000319
13	0.000052	0.000360	0.000344	0.000301
14	0.000410	0.000267	0.050445	0.000169
15	0.000084	0.000082	0.000115	0.000511
16	0.000641	0.000395	0.000412	0.000102

The CAVs for the ANLFSR and EANLFSR schemes were selected from the values given in [15] for the case of 16-bit accumulator, while they were randomly selected for the case of 24-bit accumulator, because there are no satisfying values in that case.

Each entry in Tables II–IV gives the smallest number of test vectors required to achieve 100% single stuck-at fault coverage, or the fault coverage obtained after applying 64 000 patterns to the corresponding CUT, or the fault coverage obtained along with the required number of test vectors when more than one scheme achieves the same fault coverage. The best results obtained for each benchmark circuit are shaded.

Tables II and III clearly show that the bit sequences produced by the proposed scheme in most cases outperform both the LFSR and the other arithmetic-module-based TPGs. The increase of the accumulator size has clearly resulted in certain improvements in the obtained fault coverage and, in many cases, has led to a considerable reduction of the required test length to achieve 100% fault coverage. Additional experimental results for the three more efficient schemes and for three generator sizes are given in Table IV (for the EANLFSR and $K = 32$, no CAV yielding a close-to-maximal length sequence was available). The conclusions are similar.

Apart from the achievable fault coverage and the required test-sequence length, another feature is very significant, which is the effort required for finding an efficient test sequence, with respect to fault coverage and its length. To have an estimation of the required effort, we give in Table V the maximum, the average, and the standard deviation of the fault coverage over the 20 test sequences generated for the case $K = 17$ in our experiments. From Table V, we can easily see that the standard deviation of the fault coverage for the test sequences generated by the proposed scheme, an LFSR, or MAC [10] is very small, while for the rest of the arithmetic-function-based methods [11], [12] is significantly larger. From Table V, we can also see that in most cases, the average fault coverage obtained by the test sequences generated by the proposed method is larger than the average fault coverage obtained by other schemes. Therefore, we conclude that the effort required for finding an efficient test sequence generated by the proposed method is small.

Finally, as an indication of the randomness quality of the generated bit sequences, we applied the Cesaro test [19] for the computation of $6/\pi^2$. Table VI shows a sample of the values of the absolute deviations from the correct value of π obtained for $K = 28$ for the LFSR (with primitive trinomial), ANLFSR, EANLFSR, and ACC-XOR schemes. The bit sequence was taken each time from the $(K - 1)$th bit while the bit width of the numbers considered in the test ranged from 8 to 16. As can be observed, the proposed scheme exhibits in general good behavior even on this metric.

IV. Conclusion

In this paper, we have shown that an accumulator-based TPG can be used efficiently for bit-serial test-pattern generation and that it compares favorably to LFSR and other arithmetic-function-based bit-serial sequence generators. The proposed scheme also constitutes an example of a nonlinear TPG where maximum-length sequences can be found without resorting to the brute-force approach.

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 & Mixed Signal 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] ——, "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 pseudo-random test vectors," in *Proc. Eur. Design and Test 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] D. Bakalis, D. Nikolos, and X. Kavousianos, "Test response compaction by an accumulator behaving as a multiple input non-linear feedback shift register," in *Proc. IEEE Int. Test Conf.*, Atlantic City, NJ, Oct. 2000, pp. 804–811.

[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] E. Kalligeros, X. Kavousianos, D. Bakalis, and D. Nikolos, "On-the-fly reseeding: A new reseeding technique for test-per-clock BIST," *J. Electron. Test.—Theory Appl.*, vol. 18, no. 3, pp. 315–332, Jun. 2002.

[15] G. Dimitrakopoulos, D. Nikolos, and D. Bakalis, "Bit-serial test pattern generation by an accumulator behaving as a non-linear feedback shift register," in *Proc. IEEE Int. On-Line Testing Workshop*, Bendor, France, Jul. 2002, pp. 152–157.

[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] D. Bakalis, D. Nikolos, H. T. Vergos, and X. Kavousianos, "On accumulator-based bit-serial test response compaction schemes," in *Proc. IEEE Int. Symp. Quality Electronic Design*, San Jose, CA, Mar. 2001, pp. 350–355.

[19] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Reading, MA: Addison-Wesley, 1997.

[20] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. IEEE Int. Symp. Circuits Systems*, Portland, OR, May 1989, pp. 1929–1934.

Low-Power BIST With a Smoother and Scan-Chain Reorder Under Optimal Cluster Size

Nan-Cheng Lai, Sying-Jyan Wang, and Yu-Hsuan Fu

***Abstract*—The authors propose a low-power testing methodology for the scan-based built-in self-test. This approach combines a low-power test pattern generator (TPG) with scan-chain reordering to achieve low-power testing without losing fault coverage. Three main issues are addressed. First, a smoother is included in the TPG to reduce the average power consumption. However, the fault coverage may be adversely affected by the smoother; hence, a cluster-based scan-chain reordering is employed to remedy this problem. If a very-large power reduction is necessary, the fault-coverage drop can become significant. This can be addressed by reseeding. The second topic of this paper is to give a detailed analysis on the optimal cluster size to minimize the scan-chain length. Finally, a fast and efficient algorithm is developed for scan-chain reorder in order to improve the fault coverage. The reordering algorithm is very efficient in terms of computation time, and the routing length of the reordered scan chain is comparable to or smaller than the result given by commercial tools. Experimental results show that the proposed method provides a significant and consistent reduction in the average test power, and the fault coverage is similar to previous methods with the same test lengths.**

***Index Terms*—Built-in self-test (BIST), design for testability, low-power design, routing, testing.**

I. Introduction

Predesigned intellectual-property (IP) cores are commonly used in system-on-chip (SOC) designs. Due to the large number of IP cores and the limited pin count, the long test time required becomes a great challenge for SOC testing. The built-in self-test (BIST) is an attractive alternative to SOC testing. This technique can conduct at-speed testing, and it alleviates the burden of external testers. However, the existence of random-pattern-resistant faults often causes an unacceptably long test sequence to attain high fault coverage. Several techniques have been proposed to address this problem, including Markov-source BIST [1], [2] and reseeding [3].

Another serious problem associated with pseudorandom BIST is the higher power consumption [4]. Most contemporary designs employ low-power design techniques to reduce power dissipation during normal operation. The power constraints defined under normal operations are usually much lower than the power consumed in the test mode [5]. Techniques for average power reduction will help prevent unwanted test failures. In order to perform a nondestructive test, we have to satisfy power constraints defined in the design phase. Several techniques have been proposed, including low-power BIST [6]–[8] and minimizing power during scan testing [1], [2], [9]–[11].

Scan-based BIST architectures are popular because of their low impact on area and performance. However, they also generate excessive heat dissipation in the scan-shift operations. We propose an approach to reduce the average power consumption during scan-shift operations without degrading the normal-mode performance. The proposed test

Manuscript received February 15, 2005; revised May 23, 2005 and August 26, 2005. This work was supported by the National Science Council under Grant NSC 91-2215-E-005-004. This paper was presented in part at the Asian Test Symposium, Kenting, Taiwan, November 2004. This paper was recommended by Associate Editor S. M. Reddy.

The authors are with the Department of Computer Science, National Chung-Hsing University, Taichung 402, Taiwan, R.O.C. (e-mail: sjwang@cs.nchu.edu.tw).

Digital Object Identifier 10.1109/TCAD.2006.870861