Cellular automata circuits for built-in self-test

by P. D. Hortensius
R. D. McLeod
B. W. Podaima

Results are presented for a variation on a built-in self-test (BIST) technique based upon a distributed pseudorandom number generator derived from a one-dimensional cellular automata (CA) array. These cellular automata logic block observation (CALBO) circuits provide an alternative to conventional design for testability circuitry such as built-in logic block observation (BILBO) as a direct consequence of reduced cross-correlation between the bit streams which are used as inputs to the logic unit under test. The issue of generating probabilistically weighted test patterns for use in built-in self-test is also addressed. The methodology presented considers the suitability of incorporating structures based on cellular automata, a strategy which, in general, improves test pattern quality. Thus, CA-based structures qualify as attractive candidates for use in weighted test pattern generator design. The analysis involved in determining and statistically evaluating these potential models is discussed, and is compared with that for previous as well as statistically independent models. Relevant signature analysis properties for elementary one-dimensional cellular automata are also discussed. It is found that cellular automata with cyclic-group rules provide signature analysis properties comparable to those of the linear feedback shift register. The results presented here are based upon simulation.

Introduction

Design for testability (DFT) techniques attempt to deal with the complexity of the VLSI testing problem by incorporating testability as a primary component of the design process [1]. A common feature of DFT techniques is the reconfiguration of a sequential circuit so that at test time it can be considered combinational. This is accomplished by using the sequential circuit latches to apply appropriate test vectors and accumulate the resulting response vectors. The latches are tested indirectly as they verify the combinational logic of the circuit under test. Level-sensitive scan design (LSSD) [2] is an example of such an approach.

In LSSD and similar approaches such as Scan Path [3], Random Access Scan [4], and Scan/Set [5], a test set must be determined, together with the valid responses, in advance of the test. At test time each test vector must be

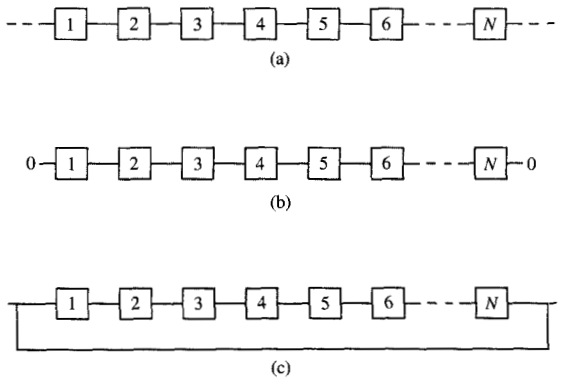


Figure 1

(a) A simple one-dimensional cellular automaton. (b) Null boundary conditions. (c) Cyclic boundary conditions.

serially scanned into the circuit, and the corresponding response serially scanned out. While this type of approach greatly reduces the complexity of sequential circuit testing, there are three difficulties:

1. An appropriate test set must be determined, which can require significant computation.
2. The time required to scan the test vectors in and the circuit responses out can be excessive.
3. The correct responses must be stored and compared to the observed responses to determine whether there is a detected fault.

Built-in self-test (BIST) techniques attempt to address these points. In a BIST design, the generation and application of the test vectors and the analysis of the resulting response are part of the circuit (or system) under test. As in scan-path techniques, a sequential circuit is rendered combinational, with the sequential circuit latches used as an integral part of the test.

A BIST design requires a mechanism for generating an appropriate set of test vectors. For cases where an exhaustive test set is prohibitive, a pseudorandomly selected subset of the possible inputs to the circuit under test is used. This requires an on-chip pseudorandom sequence generator which, in order to reduce the overhead required for BIST, should consist largely of the sequential circuit latches. An example of such a technique is built-in logic block observation (BILBO) [6], which typically employs a linear feedback shift register (LFSR) with maximal cycle length as the pseudorandom sequence generator.

The LFSR-based test pattern generator is formed by the addition of *exclusive-or* gates to the sequential latches with appropriate control logic so that the latches can perform their normal circuit function as well as be reconfigured for testing. The positioning of the *exclusive-or* gates is given by the primitive polynomial over GF(2) required to form a maximal-cycle-length LFSR [7]. A potential difficulty is the requirement of a feedback path from the most significant to the least significant cell in the LFSR, which further complicates the layout of the register and in some cases may degrade performance.

New pseudorandom number generators (PRNGs) based on cellular automata (CA) are discussed and examined using the same metrics as those for the LFSR. It is shown that these CA-based generators provide an alternative to conventional LFSR-based generators. In addition to improved randomness properties, these new pseudorandom test pattern generators can be designed to require only adjacent neighbor communication.

Also discussed is weighted test pattern generation using CA-based driving engines. The concept of weighted test pattern generation was first initiated to increase the detectability of hard-to-detect or pseudorandom-test-pattern-resistant faults [8–10]. By weighting the input probability distribution, an attempt is made to expose the hard-to-detect faults, thereby rendering them random-pattern-testable. In other words, an optimal input probability distribution is desired in order to maximize fault coverage and minimize test length. (This is basically an extension of earlier work concerning equiprobable, unbiased, pseudorandom test patterns, with the single stuck-at fault model assumed in the derivation of test quality measures.) Overall, there exists a class of combinational networks whose testability may be significantly improved by utilizing a weighted probability distribution.

BIST also requires a mechanism for reducing the response data to a simple pass/fail result using some form of data compaction. Once again, the common suggestion is to employ an LFSR to form a signature for the output data. The use of a CA-based signature register instead of one based on an LFSR would then be a natural extension in a CA-based BIST scheme. Analysis of the effectiveness of some CA-based data compactors [11] indicates that aliasing properties comparable to those of the LFSR are possible.

Cellular automata

A cellular automaton evolves in discrete steps, with the next value of one site determined by its previous value and that of a set of sites called the neighbor sites. The extent of the neighborhood can vary, depending among other factors upon the dimensionality of the CA under consideration. **Figure 1** illustrates three simple one-

dimensional CAs, where the next value at a site depends only on its present value and the values of the left and right neighbors [Figure 1(a)]; the CA register may possess null boundary conditions (i.e., the first and last sites consider their missing neighbor site to always have a zero value) [Figure 1(b)]; or the CA register may be cyclically connected (i.e., the CA forms a ring, thereby making the first and last sites neighbors) [Figure 1(c)]. Here, only binary one-dimensional CAs with two neighbor sites (left and right) are considered, but in general it is possible to use any desired modulus, dimension, or neighbor set. For a binary CA of this type, each site must determine its next value on the basis of the eight possible combinations of its own present value and those of its left and right neighbors (i.e., 000, 001, 010, $\cdots$). The next-state values corresponding to each possible input form a number which is referred to as the "rule number" under the classification scheme of Wolfram [12]. As an example, for the CA rule 90 (see Table 3, shown later), the next value of a site is the sum modulo 2 of its neighboring sites. The evolution of a CA is often shown using a state-time diagram, as in **Figure 2**, which shows the evolution for a rule 90 CA with 17 sites, for 40 time steps. The state-time diagrams presented in this paper show the evolution of numbers in the CA (or LFSR) by assigning each bit in the CA to a horizontal pixel and assigning the pixel black if the corresponding bit is a logical 1. The time axis runs vertically, thereby showing successive values in the CA. Therefore, in Figure 2 the first number is a single-bit logical 1 in the middle of the CA. On the next time step, there are two bits of value 1 to the left and right of the original bit, which is now 0. This continues with a new line for each time step of the LFSR or CA. There are in general at least two distinct methods of initializing a CA. One method is to begin with a simple state such as a nonzero value at a single central site; the other method is to begin with each site randomly initialized to 0 or 1 with $p(0) = p(1) = 0.5$. Figure 2 was initialized with a single nonzero site.

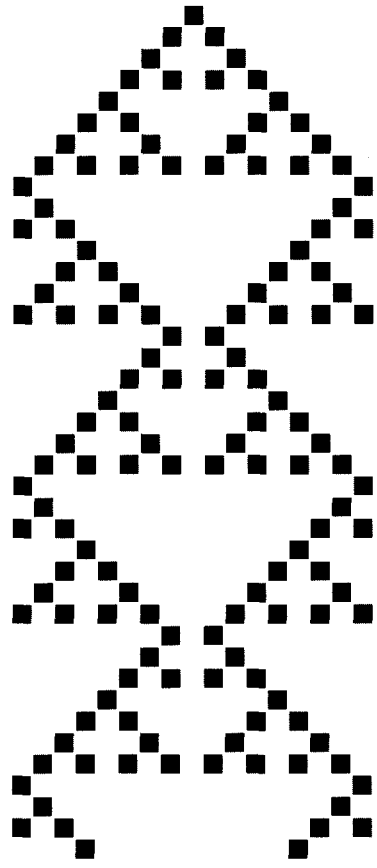


Figure 2

State-time diagram for CA rule 90 of 17 sites, for 40 time steps.

Pseudorandom test pattern generation for BIST

• *LFSR-based pseudorandom sequence generators*

The most popular hardware pseudorandom sequence generator is the linear feedback shift register. The binary sequence at cell i is generally considered to display attributes of a pseudorandom binary sequence. The sequence has a cycle length of $2^n - 1$ using an n-bit shift register, provided the polynomial describing the register is primitive over GF(2) [7]. Here we consider the two most popular methods for generating pseudorandom sequences using LFSRs. The serial-in parallel-out method forms m-bit pseudorandom numbers by collecting m bits in sequence from bit i in the LFSR. This means that it takes m shifts, or clock cycles, to form the pseudorandom number. To overcome this time penalty (the m shifts) and area penalty (the m-bit register), the bits of the LFSR are sometimes used in parallel, so that a new pseudorandom word is formed on each clock cycle. This is the method used extensively in the application of LFSRs to built-in self-test circuits [1].

Testing of these two types of LFSR-based generators using standard random-number tests shows that the serial-in parallel-out method provides good m-bit-word pseudorandom sequences. However, the parallel LFSR method (using any number of different primitive polynomials) does not yield output sequences which could be considered pseudorandom [11, 13][1]. This can

[1] It is possible to use an XOR network on the LFSR outputs to perform a transform in one clock cycle of the initial state of the LFSR to that which would occur after m shifts [13]. We do not consider this method here because of the size and complexity of the XOR network and the reduced cycle length of the output sequence.

Figure 3

State-time diagram for a parallel LFSR.

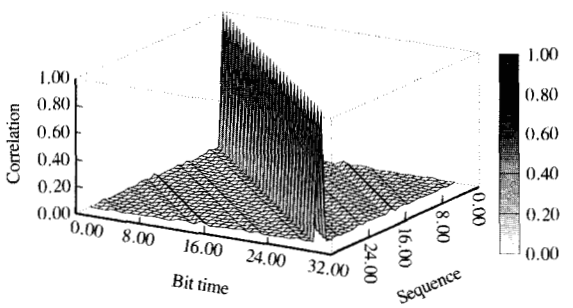


Figure 4

The cross-correlation of bit sequences in the parallel LFSR method.

readily be seen in the state-time diagram of the parallel LFSR implementation given in **Figure 3**, where there is considerable regularity in the parallel LFSR output pattern. The only criterion for pseudorandomness met by the parallel LFSR method is the equidistribution test.

The most evident failure of the parallel LFSR method is in the bit-sequence correlation (this is the feature which creates the stripes in Figure 3). A correlation figure such as **Figure 4** can be used to show both the autocorrelation and cross-correlation of bits in word sequences produced by the parallel LFSR. Here the correlation figures display the results for 30-bit words using a three-dimensional figure. The vertical axis is the magnitude of the correlation, while the x and y axes give the time displacement, i.e., number of shifts, and sequence displacement, i.e., number of LFSR cell positions, from the reference time and sequence, respectively.

The parallel LFSR method displays a severe correlation problem. In fact, the bits in the bit streams are perfectly correlated in that the value at bit i at time t will appear at bit $j > i$ at time $t + (j - i)$. Therefore, one cannot consider the test pattern sequences used in most built-in self-test structures to be pseudorandom, since the bits in succeeding test patterns are fully correlated.

The cross-correlation of the bit streams in the LFSR yields a number of circuit faults which cannot be detected. For example, a simple CMOS NAND gate with inputs A and B has faults which cannot be detected by LFSR-based test patterns. If we assume an open circuit fault on the B-input p transistor, we induce memory into the circuit, since the input $A = 1$, $B = 0$ results in a floating output. This situation, in which the last output value is held, can only be detected by having the input pattern 10 follow input pattern 11. However, this situation can never arise in LFSR-based testing if the shift direction is from A to B, since the value on input A will be on B when the next input pattern is applied. Therefore, one can never completely test a simple two-input CMOS NAND gate for stuck-open faults using single-clocked parallel LFSRs. This shows the potential deficiency of LFSR-based test pattern generation for BIST due to the correlation between adjacent outputs.

• *CA-based pseudorandom sequence generators*

A number of CA-based pseudorandom sequence generators are examined in [14]. Here we briefly describe two interesting CA-based PRNGs.

Rule 30 CA

Consider a simple one-dimensional CA using rule 30; i.e.,

$$a_i(t+1) = a_{i-1}(t) \oplus [a_i(t) \cup a_{i+1}(t)]. \quad (1)$$

The rule 30 CA displays many attributes of a pseudorandom number generator when connected in a cyclic configuration. Specifically, the rule 30 CA has significantly reduced cross-correlation as compared to the parallel LFSR generator. The principal difficulty with respect to BIST is that the rule 30 CA is not a maximal-length sequence generator. The state transition diagram is actually made up of trees and cycles. An important consideration in the use of any PRNG is the length of the sequence produced (i.e., after how many numbers does the sequence repeat?). The LFSR generators of interest here have sequence length of $2^n - 1$, where n is the length of the LFSR. However, the rule 30 CA does not provide nearly as long a sequence. For rule 30, as the length of the CA increases, the maximum possible cycle length of the pseudorandom sequence generally increases, but this growth is not monotonic; in addition, the initial state

used in the CA affects the length of the sequence produced [14, 15].

Rule 90 and 150 hybrid CA

A cellular automaton which yields a maximal-length binary sequence from each site (i.e., $2^n - 1$), like the maximal-length LFSR, is a *hybrid CA* (HCA) [16]. The HCA combines rules 90,

$$a_i(t + 1) = a_{i-1}(t) \oplus a_{i+1}(t), \quad (2)$$

and 150,

$$a_i(t + 1) = a_{i-1}(t) \oplus a_i(t) \oplus a_{i+1}(t). \quad (3)$$

The ordering of the rules for construction of a maximal-length binary sequence is irregular, with complexity similar to that involved in determining the polynomial for a maximal-length LFSR [17]. **Table 1** gives a sample of hybrid constructions for producing HCAs with maximal cycle length up to length 53. Here, 1 refers to CA rule 150. Hence, a length-5 maximal-length hybrid would be constructed by having CA rules 90 and 150 in the following order: 150, 150, 90, 90, 150. It should be noted that for many lengths there are several CA rule 90 and 150 hybrid constructions which will yield maximal-length cycles. Maximal-cycle-length hybrid cellular automata exist for lengths larger than 53 but must be found using computer simulation.

Zhang et al. [18] have recently developed an efficient algorithm to generate minimum-cost maximal-length HCAs; at the time of writing, they have produced a table giving constructions up to length 150. Using this type of table, construction of the HCA is quite simple. For example, consider a HCA of length 16. A maximal-length cycle can be formed by twinning CA rule 150 at one end of the automaton and then alternating rules 90 and 150 over the rest of the CA. **Figure 5** shows the state-time diagram for a 498-site hybrid over 840 time steps with a simple initial state. Note that the regular pattern dies out as the CA evolves in time.

Unlike the LFSR or even the rule 30 CA, adjacent sites in the HCA are not correlated in time or space[2], as is evident in the autocorrelation and cross-correlation data of **Figure 6**.

The HCA with null boundary conditions has layout advantages over rule 30, since the first and last sites in the hybrid need not be connected (i.e., no extended wiring is required).

Weighted test pattern generation for BIST

Because of the versatility with which pseudorandom patterns may be distributed in a system and the strikingly high fault-detection capability of these test patterns, pseudorandom pattern testing is particularly suited for use in a BIST strategy. Today, there is little argument that this ideology is applicable to a large class of combinational circuits. Often, however, there are instances where the testability of a circuit is hindered by a relatively small number of hard-to-detect faults. When ordinary unbiased pseudorandom testing is performed, these faults, categorically known as pseudorandom-resistant faults, display very low fault-detection probabilities. Circuits known to contain such faults are called *pseudorandom-resistant*; they must be dealt with by an advanced form of test.

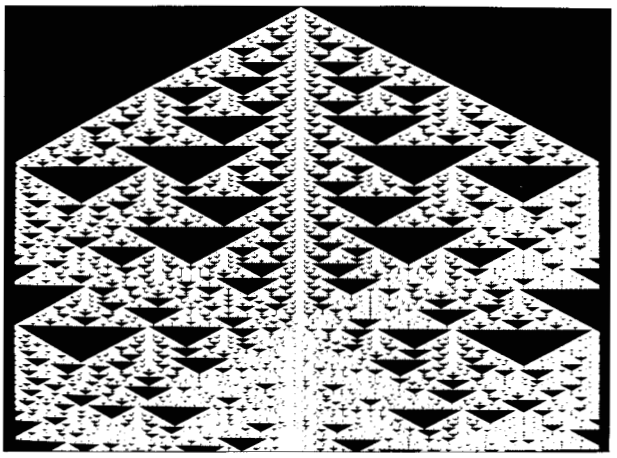


Figure 5

State-time diagram for a 498-site hybrid CA, for 840 time steps. Null boundary conditions; initial state with a single nonzero site.

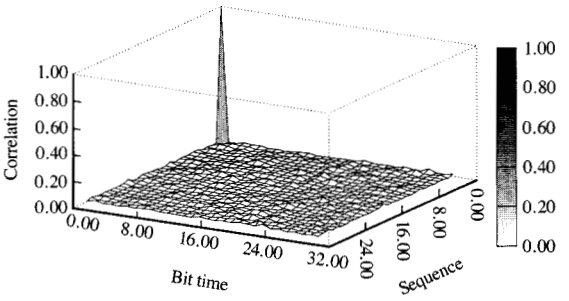


Figure 6

Cross-correlation of site values in a 30-site hybrid CA.

[2] For the 90/150 maximal-length HCAs, the parallel bit streams are actually the same (true pseudonoise sequences), except for some unknown phase or time offset [19]. For test applications the parallel bit streams are effectively uncorrelated.

Table 1 Hybrid constructions necessary to achieve a cellular automaton with maximal cycle length.

Length n	*Construction*	*Cycle length*
4	0101	15
5	11001	31
6	010101	63
7	1101010	127
8	11010101	255
9	110010101	511
10	0101010101	1,023
11	11010101010	2,047
12	010101010101	4,095
13	1100101010100	8,191
14	01111101111110	16,383
15	100100010100001	32,767
16	1101010101010101	65,535
17	01111101111110011	131,071
18	0101010101010101	262,143
19	0110100110110001001	524,287
20	11110011101101111111	1,048,575
21	01111001100000111011	2,097,151
22	0101010101010101010101	4,194,303
23	11010111001110100011010	8,388,607
24	11111101001011010101011 0	16,777,215
25	10111101010101001111001 00	33,554,431
26	01011010110100010111011000	67,108,863
27	000011111000001100100001101	134,217,727
28	0101010101010101010101010101	268,435,455
29	10101001010111001010001000011	$2^n - 1$
30	11101000100111011001010001111101	$2^n - 1$
31	01001100101011011111101110011000	$2^n - 1$
32	01000110000010011011101111010101	$2^n - 1$
33	00001100010011100111001011000 0101	$2^n - 1$
34	00111100001011010000110001101 11010	$2^n - 1$
35	01010111101111011001110101001010011	$2^n - 1$
36	10100110010010001111101011000010 0011	$2^n - 1$
37	00100101100111101011010110000010110011	$2^n - 1$
38	00011100101011110110011001111000010011	$2^n - 1$
39	11010001011111011011110011001110110 1100	$2^n - 1$
40	00001110110010101011111001000010111001 01	$2^n - 1$
41	01101011111110100001011001100011110000111	$2^n - 1$
42	00100111111011001110010100100110011110 0110	$2^n - 1$
43	00111010111000101110001000010110101100 10010	$2^n - 1$
44	00111100111101110101101110000100101011000010	$2^n - 1$
45	00110100101100110110100100010011000110100 1101	$2^n - 1$
46	00010010100110010100011010001011001110110 10110	$2^n - 1$
47	00111001011111100111001010100100010111000001101	$2^n - 1$
48	00011000011011111001001010011101000111100000 1111	$2^n - 1$
49	00101101111011001000110010111110001011101100 11001	$2^n - 1$
50	10011010011011000000110001101000101100100010010110	$2^n - 1$
51	00010000101110101010000010110100111010001010000 10111	$2^n - 1$
52	00110010001101111011011111110001000111101011100 0110	$2^n - 1$
53	10000111001010001000001001001100101110111110110010101	$2^n - 1$

A successful candidate, which maintains many of the attributes first introduced by unbiased pseudorandom test pattern generation, is weighted test pattern generation (WTPG). This approach involves the application of weighted probability distributions in order to enhance the probability of generating suitable patterns to expose the pseudorandom-resistant faults [10]. In doing so, the overall test set decreases while the fault coverage increases, rendering this particular subclass of circuits weighted-random-pattern-testable. Work by Wunderlich [20] and by Waicukauski and Lindbloom [21] has successfully demonstrated that this is, in fact, the case for the ISCAS [22] benchmark designs.

By using an adaptive weighted test pattern generation method or a probabilistic fault-grading technique, it is possible to find the relationship between the fault detection and rate of excitation of a circuit. With this knowledge, an appropriate weighted probability distribution(s) may then be assigned. Although more recent advances in WTPG have resulted in some efficient

heuristics for computing multiple weighted test sets, earlier work of Chin and McCluskey [10] suffices to demonstrate, in a simplified manner, the attributes of WTPG.

• *Cellular-automata-based WTPG considerations*

Here we consider using driving engines with external weighting logic for realizing cellular-automata-based WTPGs. An alternative technique would be to use conventional and hybrid cellular automata directly.

Past methods of WTPG design incorporate the maximal-length LFSR as a primary "driving engine," supplying pseudorandom patterns to an array of logic gates performing a weighting operation on incoming patterns. Naturally, it is expected that if a more capable pseudorandom, or even random, number generator is used, the result will be an improvement in the statistical properties of the WTPG function. For example, if measures of effectiveness are obtained for a statistical distribution of independent inputs, the correlation between expected and observed measures will be greater using an improved WTPG. For the purpose of achieving different output probabilities, standard Boolean logic gates, each with its own unique probability profile, may be configured to deliver the desired weighting to each input of the circuit under test.

By connecting various logic gates, forming single-level or multi-level logic arrays, a number of incremental output probabilities are attainable. For example, if a 50% equiprobable driving engine is used, only probability increments of 1/4 are possible by incorporating a single-level array of logic. However, if a two-level array of logic is utilized instead, probability increments of 1/16 are permitted. Even though still finer increments are possible, it should be kept in mind that the number of usable outputs decreases substantially as the number of logic levels increases. For this reason, the resolution of the probability increment is limited by the restrictions imposed upon it by VLSI—in particular, the area constraint. Recently, Brglez et al. [23] have shown that multiplexors may also be used to accomplish much the same task. In addition, the multiplexor technique using CA outperformed the LFSR-based WTPGs with respect to fault coverage for their application.

For our purposes here, we are primarily concerned with the HCA as well as with LFSR driving engines. In [19] four different driving engines were investigated. They included a nonlinear random number generator[3], a pseudorandom maximal-length LFSR, a pseudorandom maximal-length rule 90/150 HCA, and a pseudorandom nonmaximal-length rule 30 CA. With the exception of the pictorial state-time evolutions, where the original

[3] The nonlinear random number generator is based on a uniform distribution generated by a nonlinear additive feedback shift register (NLFSR).

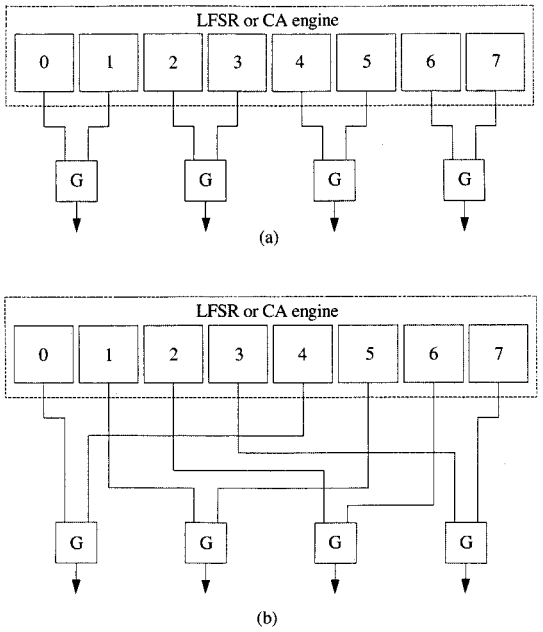


Figure 7

Basic parallel WTPG configurations: (a) zero spacing configuration [$J = 0$]; (b) $N/2$ configuration [$J = (N/2) - 1$].

driving engines are 53 cells wide (to allow for better visual presentation), all statistical analysis is performed on machines based on engines 30 cells wide (because they are large enough to exhibit global characteristics, but small enough to allow extensive analysis). Complete details regarding statistical evaluation and the statistical estimators used can also be found in [19]. For the HCA of width 30, the combination

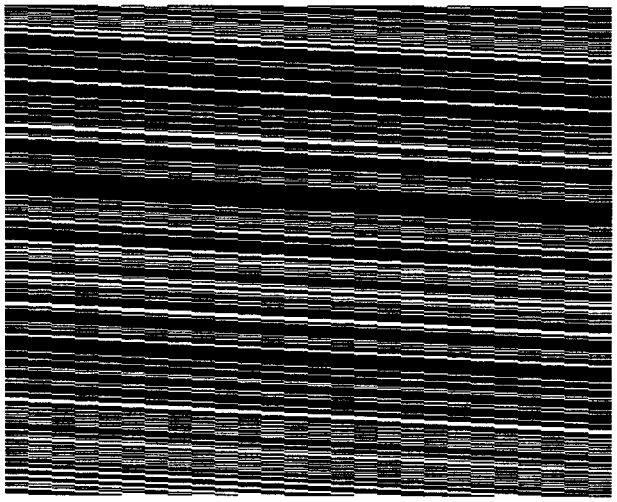
Rule 90/150 HCA:

$$000001100010000110000100111110 \qquad (4)$$

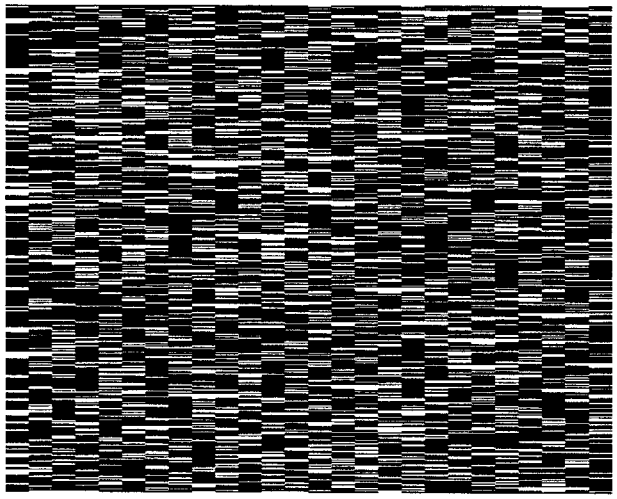
is used, where a "0" represents a rule 90 cell and a "1," a rule 150 cell.

• *WTPG logical configurations*

Several interconnection schemes are feasible, with varying degrees of complexity and area overhead. A basic configuration is characterized by the amount of cell spacing between each consecutive pairing of gates, for a particular logic level. These basic configurations include a zero-spacing configuration [$J = 0$] and an $N/2$ configuration [$J = (N/2) - 1$] (see **Figure 7**). Each logic level may contain a different basic configuration, but it is



(a)



(b)

Figure 8

Space-time evolution of test patterns for (a) $N/2$ LFSR and (b) $J = 0$ HCA weighted to 75%.

the overall logic array itself, made up of one or more logic levels, which determines the final output probability assignments. The selection of an appropriate interconnection scheme (logic array) for a particular driving engine depends on the statistical qualities demanded, and the conformity of its implementation to VLSI. These factors help establish a criterion to make possible the proper selection among the different WTPG candidates. For our purposes, we focus our discussion still further, considering the $N/2$ LFSR and $J = 0$ HCA configurations, which were selected because they represent the most likely candidates for practical WTPG circuits. Complete characterization of all configurations can be found in [19]. For most of the following discussion, our interest is in the generation and analysis of weighted broadside words with the same weight at each bit position. This facilitates the assignment and analysis of random variables and the abstraction of statistical measures. Individual bit-stream tests are also included for completeness. It should be noted that the following discussion does not preclude the application of CA pseudorandom pattern generators with a different weight at each bit position or group of bits. Statistical attributes are preserved due to the autonomous nature of the weighting gates and the apparent statistical independence of their inputs.

- *State-time visualization*

A simple apparatus which is often used, albeit with caution, because of its "unequivocal" ability to immediately assess the randomness of an emanating process, is the human eye. As is well appreciated, the eye sometimes fails to interpret accurately what actually appears. Keeping this in mind, a strictly qualitative visual test, with questionable validity, may be performed on a variety of finite-state-machine state-time evolutions. **Figure 8** shows the state-time evolutions of the weighted LFSR and HCA driving engines. Throughout the discussion we consider 75% 1s weighting for illustration (an OR array performs a 75% weighting).

By searching for some obvious patterns within an evolution, important pictorial comparisons can be made. Generally speaking, the appearance of large global, as opposed to local, patterns is probably damaging to a system's subjective randomness. The test itself can thus be used as a weak pre-test for random behavior. The figure shows noticeable global nonrandomness for the LFSR-based configuration; for those based on cellular automata, only local self-similar structures prevail. For a more quantitative look into random performance, much more sophisticated objective analysis is needed.

- *Density and average density*

The density, d_T, of a binary word (test pattern) is defined here as the actual number of ones per word length, W, or

$$d_T = \frac{n_1(T)}{W}, \tag{5}$$

where n_1 is the number of 1s; the average density,

$$D_T = \frac{1}{T+1} \sum_{i=0}^{T} d_i, \tag{6}$$

is merely the average of the densities for some time evolution $T \in \{0, \cdots, L-1\}$, where L represents the total number of words in an evolution. The densities emanating over time reveal information about the

sophistication of the underlying process. For instance, if a system possesses total statistical independence, the L random variables making up the time evolution are mutually independent and identically distributed (IID). Thus, graphically, the density evolution will appear similar to band-limited white noise. Also, because the statistical characteristics of a discrete independent temporal process do not change with time (i.e., it is said to be a discrete-time stationary process), the average density evolution converges rapidly to some mean, μ, with infinitesimal steady-state error.

Figure 9 illustrates the average density evolutions of the $N/2$ LFSR and the $J = 0$ HCA. These original finite-state machines each consist of 30 sites, and, as indicated previously, are used as a primary source of pseudorandom test patterns which feed weighting logic arrays. For the various gate configurations, the ability of the original generator to produce high-quality pseudorandom numbers is critical to the generation of high-quality weighted random test patterns.

By observing average density evolutions of the various structures in their different configurations, it is apparent that the configurations of the cellular-automata-based structures are attractive in that they resemble more closely the evolution based on a statistically independent model.

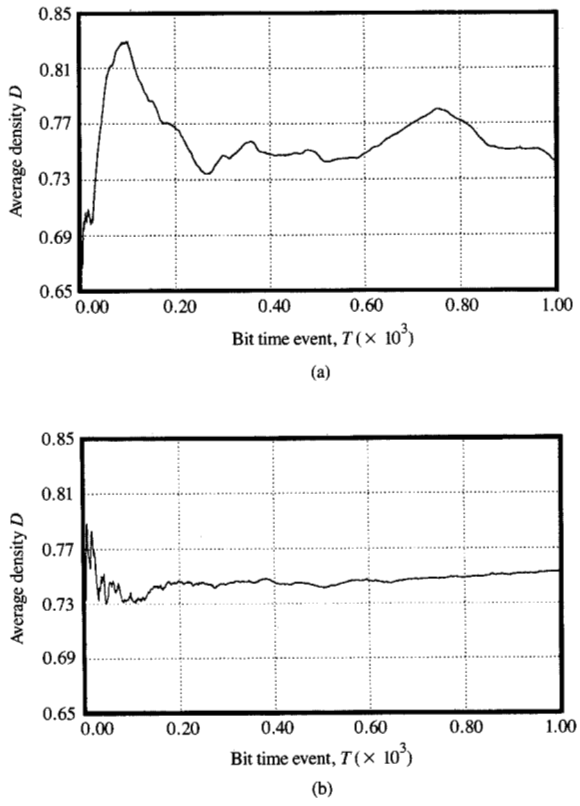


Figure 9

Average density evolution of test patterns for (a) $N/2$ LFSR and (b) $J = 0$ HCA weighted to 75%.

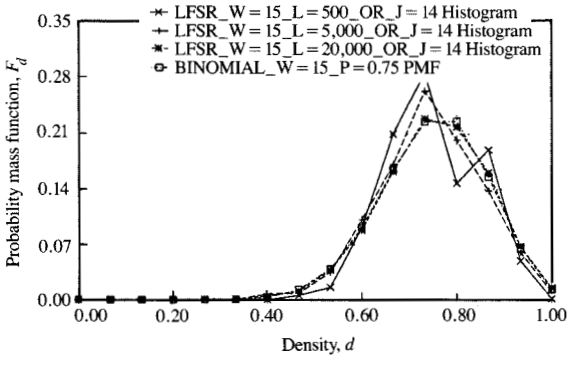
- *Probability mass function*

Any *discrete-time stochastic process* can, in part, be described by its discrete probability density function, or probability mass function (PMF). This emphasizes the importance of deriving a histogram which, at least approximately, represents the PMF of some random process. For any finite-state machine, each site position can take on only one of two values, "0" or "1." If it is further stipulated that the process governing the evolution of a finite-state machine be statistically independent, each bit position can be completely specified by a *Bernoulli* random variable. When the entire word (width) is examined, a function of W Bernoulli random variables can be formulated contributing to the specification of the entire finite-state machine. For such a system, the densities d_T emanating over time conform to a discrete *binomial or delta* PMF. It is then possible to compare a histogram generated empirically, by some deterministic finite-state machine, to a PMF obtained assuming total statistical independence. Although this comparison does not provide information as to how independent, or dependent, a particular deterministic process is, it does reveal some important aspects of distribution convergence. Naturally, it is expected that a machine based on a statistically independent model will converge rather rapidly to a binomial PMF.

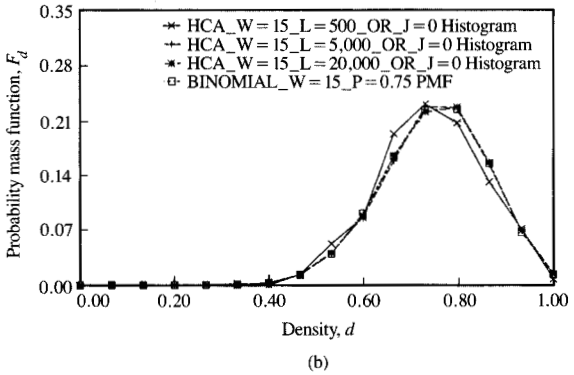
Figure 10 represents the density histogram evolutions of the $N/2$ LJSR and the $J = 0$ HCA configurations at 75%. It is evident from these plots that the HCA-based structure converges more quickly and is thus more favorable from a statistical standpoint.

- *Goodness-of-fit test*

The probability mass function, $p(X)$, or probability distribution function, $F(X)$, completely characterizes the behavior of the random variable X [24]. Because of this property, it is of great importance to establish the "goodness of fit" between a distribution (sampling distribution) or histogram determined empirically and that which is proposed or postulated. The best approach used to substantiate a measure of similarity is to test a hypothesis regarding some previously known characteristic. With the benefit of some statistical measure, a null hypothesis H_0 may at best be rejected or



(a)



(b)

Figure 10

Probability mass function (histogram) of (a) $N/2$ LFSR and (b) $J = 0$ HCA weighted to 75%.

refuted. It has been determined that a χ^2 (chi-square) statistic is particularly suited for this type of analysis. Explicitly stated, the χ^2 test is made with the null hypothesis H_0. The data $X_1, X_2, \cdots, X_L$ are IID discrete random variables, with a binomial PMF.

According to Knuth [25], the χ^2 test is perhaps the best known of all standard statistical tests. Its potential is best emphasized by virtue of the fact that it may be used in conjunction with many other statistical tests.

The χ^2 statistic is used mainly for the purpose of providing the means by which arbitrary, empirically determined, data can be compared to some ideal or "expected" value. This is done by weighting the squares of the differences between the observed data and the expected data (determined by some discrete or continuous probability function) in the form of a summation. The statistic is expressed in the form

$$\chi^2 = \sum_{i=0}^{k-1} \frac{[N_i - LP(i)]^2}{LP(i)}, \tag{7}$$

where L denotes the size of the independently selected data set, $LP(i)$ the expected number of outcomes, and N the observed number for some particular category i, ranging from 0 to $k - 1$. [For large L, Equation (7) is approximately χ^2 distributed.]

The quantity χ^2 should be reasonably small for H_0 to be refuted; otherwise, if χ^2 is considered too large, H_0 will be rejected outright. By observing tabulated quantities of a χ^2 distribution, a corresponding percentage point, or probability, is obtainable for some degree of freedom (d.f.). [As a "rule of thumb," $LP(i) \geq 5$, so the d.f. is available as a by-product of the computational process.] This percentile forms the basis of the determining factor by which the null, or alternative, hypothesis H_0 is rejected.

A more direct use of the χ^2 test, as applied to random number generator testing, involves probabilistically judging the actual data emanating from a random number generator. Although this procedure is no more difficult than that required in the testing of a null hypothesis for distribution fitting, its implications are more profound. In the view of Knuth [25], no definitive statement can be made as to whether or not a sequence is random; however, what can be stated is the probability or improbability of certain sequences being randomly generated. With respect to this outlook, he has suggested the following interpretation as a means of rating the randomness of large sequences of "apparently" random data: If χ^2 is less than 1% or greater than the 99% entry, the sequence is "rejected" as not being sufficiently random; if χ^2 resides between the 1% and 5% entries, or the 95% and 99% entries, the sequence is "suspect"; if χ^2 resides between the 5% and 10% entries or the 90% and 95% entries, the sequence is "almost suspect"; but if the χ^2 lies somewhere between 10% and 90%, the χ^2 is thought to be a value which could be produced by a random sequence.

Typically, a χ^2 test is performed at least three times on different sequences of adequate length. Also, it should be noted that for instances where χ^2 testing is applicable, it is only valid asymptotically for independently observed data. Therefore, tests which check for independence (such as a serial correlation, tuple, temporal measure entropy, spectral, lattice, or run test), depending on the kind of data comprising the sequence, should be performed first, so that the authenticity of the χ^2 can be acknowledged.

In order to make quantitative comparisons of density histogram convergence among the various configurations, a χ^2 test can be adequately performed on the histograms generated from a large test set. Because it is known that an IID statistical process dictates ideal behavior, it should, therefore, represent the postulated, or reference, function in the form of a binomial PMF. With respect to this and Equation (7), the χ^2 metric can then be

computed for an analytical establishment. Using a test set of 20 000 vectors, an approximation level between 10% and 90%, and the criterion that $LP(i) \geq 5$, **Table 2** clearly illustrates that the LFSR-based configurations generate density sequences which fail the χ^2 test, whereas most of the cellular-automata-based configurations generate density sequences which can be considered randomly generated. This overwhelmingly exemplifies the undesirability of LFSR-based WTPGs, especially considering the smaller test lengths which are more commonly associated with weighted random pattern testing.

• *Magnitude spectrum*

A measure used to indicate the rate at which densities change on a global level, within a sample function, can be made by applying a Fourier transform. Since the density sample function d_T (i.e., the density evolution of some discrete-time stochastic process) represents what can be considered a large set of points, "sampled" with fixed increments of time Δ, it is typical of what the density evolution looks like for a number of other start-up "seeds." Thus, if d_T contains L density values, its discrete Fourier transform can be determined [26] by

$$F(f_n) = \Delta \sum_{k=0}^{L-1} d_k e^{2\pi kni/L}, \tag{8}$$

where

$$d_k = d(t_k), \qquad t_k = k\Delta, \qquad k \in \{0, 1, \cdots, L-1\}, \tag{9}$$

and

$$f_n = \frac{n}{L\Delta}, \qquad n \in \left\{-\frac{L}{2}, \cdots, \frac{L}{2}\right\}. \tag{10}$$

By choosing $n \in \{0, 1, \cdots, L/2\}$, the resulting positive frequencies of a two-sided power spectral density are calculated to provide insight into the frequency content, or *magnitude spectrum* $|F_n|$ of a density time evolution. For a statistically independent process, the magnitude spectrum is expected to have an appreciably flat frequency response. **Figure 11** illustrates the magnitude spectra of the $N/2$ LFSR and $J = 0$ HCA configurations weighted at 75%.

As expected, when the magnitude spectrum of the LFSR-based configuration is observed, the majority of the power is found to be contained in the low frequencies. This trend is similar to what is expected if the originating density evolution waveforms consist of a low-frequency fundamental. This behavior is a direct result of the dependencies among consecutive density values d_T. Upon closer inspection, any succeeding density can only be the same, or differ at the most by one density increment (i.e., $d_{T+1} \in \{d_T + 1/W, d_T, d_T - 1/W\}$). On the other hand, the magnitude spectrum of the HCA

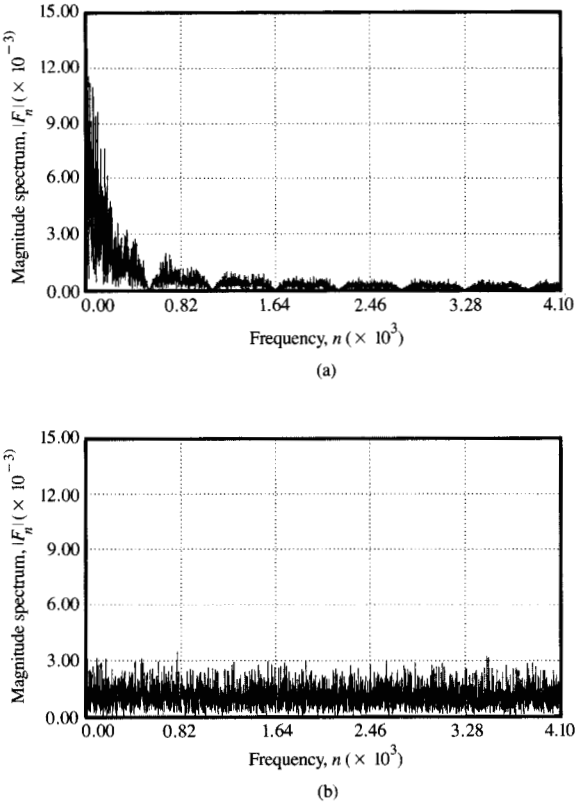


Figure 11

Magnitude spectra of (a) $N/2$ LFSR and (b) $J = 0$ HCA weighted to 75%.

Table 2 Chi-square PMF test for the various WTPG configurations. Engine refers to the original driving engine; [S] refers to a nearest neighbor sharing configuration.

Configuration	*d.f.*	*NLFSR*	*LFSR*	*HCA R90/150*	*CA R30*
Engine	18	Pass	Fail	Pass	Pass
[S]	15	Fail	Fail	Fail	Fail
[$J = 0$]	10	Pass	Fail	Pass	Pass
[$J = N/2 - 1$]	10	Pass	Fail	Pass	Pass

appears "white" or flat, as would a statistically independent model.

A serial test, which is basically a generalization of the χ^2 test to higher dimensions, is an empirical test which provides an indirect check on the assumption that individual d_Ts are independent [27]. If the d_Ts are correlated, the pairs (d_T, d_{T+1}) tend to cluster around the

diagonals of a unit square. The application of a χ^2 test will definitively detect this, and, thereby, indicate nonrandomness in the generation process. Such is the case for the density evolution of any LFSR-based configuration, where the predicting mechanism, as indicated, is rather simplistic. It is evidently the low-frequency content of the magnitude spectrum which is most damaging to randomness.

Not surprisingly, HCA-based configurations have far better suitable magnitude spectra; the HCA configurations all appear flat.

• *Correlation of weighted configurations*

As is evident from the space-phase correlation plots discussed in the previous section, the structures formed by adding different configurations of weighting logic appear to have correlations directly related to the original driving engines. The most noticeable difference is a "broadening" in some of the original correlation peaks and/or ridges.

The $N/2$ LFSR-based configuration has a broadened space-phase correlation ridge. The HCA shows no appreciable correlation peaks.

From the correlation analysis, it can be clearly stated that the CA-based configurations have more attractive state-time correlation plots than those based on the LFSR. In particular, the HCA-based configurations have the most acceptable properties of all the deterministic machines we have examined, and certainly present themselves as excellent parallel WTPG candidates. Of these configurations, the zero-spacing configuration offers a distinct advantage in the detection of stuck-open faults, beyond those based previously on the LFSR, with the added benefit of a nearest-neighbor connection scheme. Recent transition-fault analysis has demonstrated the improved coverage of CA-based TPG [28]; we believe this to be a direct consequence of the reduced cross-correlation. For this reason, the zero-spacing configuration is most recommended for implementation in a weighted-test-pattern-BIST environment. The HCA sharing configuration offers significant implementation advantages, and in spite of its increased correlation also warrants further consideration. To be more definitive, transition-fault analysis over a large number of representative circuits is still required.

• *Bit-sequence tuple lengths*

Before a bit sequence can be considered effectively random, it must possess apparent statistical independence. This is to say, besides conforming to the appropriate distribution, the emanating bits must appear as if they were generated independently. If this is the case, the number of "k-tuples" contained within a given sequence length should, by Equation (11), be relatively consistent.

The number of k-tuples, as defined here, includes the occurrence of overlapping tuples. For example, within a single run of four 1s, there are four k-tuples of length one, three k-tuples of length two, two k-tuples of length three, and one k-tuple of length four. (A so-called "run" is a contiguous sequence of 1s isolated by one or more 0s.)

The expected number of k-tuples, $T_L^{(k)}$, found in the evolution of a single Bernoulli random variable can be obtained by

$$E[T_L^{(k)}] = (L - k + 1) \cdot p^k, \tag{11}$$

where p is the probability of generating a 1 ($1 - p$ is the probability of generating a 0), L is the number of successive Bernoulli trials, and k is the tuple length. In accordance with Equation (11), an absolute, or expected, tuple profile may be used as a basis on which comparisons among a variety of machines and configurations can be made.

For a maximal-length LFSR, it is *only* for the $N/2$ configuration that the tuple profile coincides with that which is expected. These findings also happen to be independent of the chosen bit sequence, due to the fact that every consecutive bit sequence is identical, with the exception of an accompanying phase shift. The necessity of using a spacing of $J = N/2 - 1$, in order to achieve proper LFSR-generated tuple profiles, has been addressed by Chin and McCluskey [10]. They discuss the rationale behind the selection of the $N/2$ configuration for the case of a 25% weighted random bit sequence which is readily extended to the 75% case.

When the tuple profiles of a maximal-length $J = 0$ HCA configuration are examined, we find that the results are no longer completely independent of the selected bit sequence. For the $J = 0$ HCA configuration, the general trend is that the bit sequences farthest from the end sites provide the better tuple profiles. However, if rule 150 cells are used in place of the rule 90 cells at the end sites of the original generator, there is a great improvement in the tuple profile of bit sequence at the ends of the arrays. In fact, it appears as good as any of the other tuple profiles.

If the HCA is reconfigured in the $N/2$ configuration, only moderate gains are made in the tuple profiles. In light of this, and the potential advantage in overall circuit performance because of reduced wiring complexity, it is the zero-spacing ($J = 0$) configuration of the HCA which lends itself for use in BIST.

Comparison of CA and LFSR

The rule 30 and hybrid CA and the LFSR all display reasonably good spatial distribution of the pseudorandom output words, the LFSR being the least acceptable from a statistical viewpoint. The advantages of cellular automata arise from the much-reduced cross-correlation associated

with the CA compared to that of the LFSR. As discussed previously, single-bit outputs from the LFSR and CA are pseudorandom, but in most BIST applications the test patterns are generated by taking many bits of the register in parallel. This leads to nonpseudorandom sequences for the LFSR because of the correlation. Application of standard random-number tests to both the parallel LFSR and CA PRNGs shows that the CA-based PRNGs are much more pseudorandom than the parallel LFSR [11, 19]. An especially important result occurs in tests which measure the distribution of test pattern pairs, triples, and quadruples. We observe good distribution using the CA-based generators, whereas in the LFSR, pairs, triples, and quadruples are not at all well distributed.

Another concern arising from the fact that LFSRs are poor pseudorandom generators is that much of the analysis for BIST assumes a random or pseudorandom test pattern source, so when an LFSR is used as the source, it should not be expected that the fault coverage and other calculated measures will be entirely accurate. This does not imply that the fault coverage of the LFSR will be degraded, but only that the analysis in which it is considered to be random is inaccurate.

A disadvantage of some CA-based schemes is reduced cycle length and starting value dependence. For example, the rule 30 CA has maximal-length cycles which are much smaller than those of the LFSR. In addition, rule 30 is starting-value-dependent, whereas the LFSR is not (other than the zero state). The only CA discussed here with maximal-length cycles like those of the LFSR is the rule 90/150 HCA. Therefore, if cycle length is important, the LFSR or HCA should be considered. One additional point is worth noting concerning CA test circuits: We have found that the most favorable HCAs are ones constructed with a good mix of rule 90 and 150 cells.

In spite of these differences, the LFSR has proved to be an excellent BIST pattern generator on a wide variety of circuits. The somewhat surprising consequence is that important statistical attributes associated with generating good pseudorandom patterns may be of little consequence from the perspective of circuits (particularly if the single stuck-at fault model is used). At present, only limited comparisons between the LFSR and HCA WTPGs have been made. These studies indicate comparable stuck-at fault coverage and favorable stuck-open fault coverage. As we have mentioned, extensive transition-fault analysis is required.

Signature analysis

- *Traditional signature analysis*

The most popular BIST data compaction methods use error-detection and -correction techniques for cyclic redundancy check (CRC) codes. These error-detecting and -correcting circuits, which make extensive use of LFSRs, were developed in the late 1950s and early 1960s [29, 30]. They are well understood and are thoroughly explained in the algebraic coding theory literature as *syndrome detection* [31, 32] and in the digital testing literature as *signature analysis* (SA) [33, 34]. Here we focus our examination on the use of cellular automata for signature analysis in BIST.

The conventional signature analysis circuit uses a LFSR to implement a repeated polynomial division of an input binary data stream. Several well-known theorems concerning LFSR-based SA are also of particular relevance here [35–38].

As most practical circuits have many outputs, one could use a multiple-input signature analysis register (MISR) [39–41] to form a signature of the output from a multiple-output circuit. The MISR circuit is currently considered to be the most efficient means of producing a signature of a multiple-bit data stream. Several analytical measures of error-detecting capability for MISR circuits have been proposed in the literature and are again of direct relevance here. Recently, Williams et al. [38] have shown that the probability of aliasing for maximal-length LFSRs asymptotically approaches $1/(2^n - 1)$, where n is the length of the register. The analysis was based upon a single-bit-stream input to a LFSR with probability of error, p/bit.

- *CA-based signature analysis*

One of the primary motivations for considering signature analysis using cellular automata is that in cases where a CA is preferable for test vector generation, it would be desirable to use the same CA for signature analysis as well. Here we proceed to describe and propose some measures of the effectiveness of signature analysis circuits using cellular automata.

Two methods of cellular-automata-based MISR (C-MISR) implementation

The nearest-neighbor communication properties required for implementing elementary one-dimensional cellular automata allow the consideration of several different techniques of SA. Here we consider two of the most promising methods; other techniques are possible and are studied in [42]. The techniques to be considered here are shown in **Figure 12**. In Figure 12(a), the signature is formed by updating and then *exclusive-or*ing the current state at each site with the corresponding output from the circuit under test. Notice that this technique is directly analogous to those of conventional LFSR-based MISRs (L-MISRs). The second technique, shown in Figure 12(b), is similar except that here we first *exclusive-or* each site with the corresponding circuit output and then

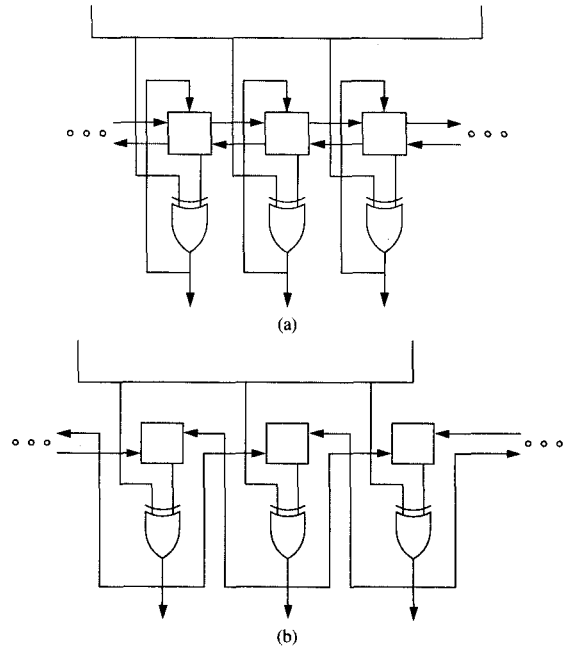


Figure 12

Two signature-analysis techniques using CA-based MISRs: (a) Method 1: $R(t+1) = R(t)' \oplus O(t+1)$. (b) Method 2: $R(t+1) = [R(t) \oplus O(t)]'$.

increment the cellular automaton. These two methods can be described algebraically by the following equations:

Method 1:

$$R(t+1) = R(t)' \oplus O(t+1); \tag{12}$$

Method 2:

$$R(t+1) = [R(t) \oplus O(t)]', \tag{13}$$

where $R(t)$ = cellular automaton contents at time t, $O(t)$ = circuit output at time t, and $R(t)'$ = incremented value of cellular automaton contents at time t.

Here some results are presented which attempt to identify properties that should be possessed by candidate cellular automata for SA, using Methods 1 and 2. The following theorems are presented without proof, since the interested reader may refer to [42] for complete development and proof of the theorems.

We first consider the evolution of states for different cellular automata. In general, the state transition diagrams consist of many cycles and paths to the cycles. These state transition diagrams contrast with the most common LFSR-based MISRs, which have one large cycle of $2^n - 1$ states and the zero state.

Theorem 1 Given a general tree structure with N_i nodes of degree i, the probability that the signatures of two random sequences differing in only one element remain distinct after T steps, using Method 1 or 2, is

$$\left\{1 - \sum_{N_i>0} \frac{i(i-1)N_i}{N^2}\right\}T. \tag{14}$$

Using Theorem 1 [42], we can now find the probability of aliasing on single-word errors for any CA-based MISR, provided we know the state transition diagram. For example, a 4-bit cyclic rule 30 cellular automaton has a state transition diagram which yields a probability of not aliasing of $[1 - (5/128)]^T = 0.961^T$. If we examine more CA-based MISRs, we see that cellular automata with the fewest branches of degree ≥2 in the state transition diagram have the smallest aliasing probability. If the state transition diagram consists only of unary branches (i.e., cycles, the aliasing probability for single-word errors is zero. **Table 3** lists a number of CA rules which lead to such behavior.

Theorem 2 Using SA methods 1 and 2, we will always have a different signature for two sequences differing in only one element, provided that the MISR's rule of operation forms a cyclic group.

Notice that Theorem 2 [42] also holds for conventional L-MISRs. Therefore, only rules implementing cyclic groups should be used for MISRs, since for single errors both C-MISRs and L-MISRs will provide the same anti-aliasing capabilities. For single error, using a C-MISR built from a CA rule of Table 3 provides SA properties equivalent to those of an L-MISR.

This can be extended to multiple errors, but the number and complexity of the terms become large. As in the single-error case, implementing SA on a general directed tree implies a greater probability of converging to the same signature than on a unary tree [11]. Once again we see that the CA rules of Table 3 are the most suitable for use in a C-MISR when we consider multiple errors.

It is possible to do empirical studies to verify our estimates on the aliasing probability for the CA rules of Table 3. One such test is to check the aliasing probability by inserting errors into a number sequence until an aliased signature occurs. These results are reported in [42]. It was found that both SA methods 1 and 2 provided essentially the same results. For comparison an L-MISR was also included. We noted that both the L-MISR and the C-MISR implementations alias at a rate which is approximately predicted by $1/(2^n - 1)$. For example, for a length 9 rule 89 cellular automaton it takes an average

of 433 quadruple error insertions before an aliased signature occurs. In addition, the simulation indicated that all the CA rules of Table 3 and the LFSR provide essentially equivalent aliasing performance for the above test.

To indicate why a C-MISR circuit may be preferable to the L-MISR circuit, we note that in an L-MISR it is possible to miss an error by canceling out the error when the LFSR is shifted [43]. If a C-MISR constructed from the rule 90 and 150 hybrid is used, the error is not canceled, since each bit in the C-MISR is a function of the incoming information and its two neighbor bits.

In general, a MISR receives a word on each cycle which is the exclusive-OR of the expected word and some, possibly all-zero, error pattern. Recent studies of the multiple-input signature analysis case by Miller [44] have shown that if the probability of occurrence is the same for all error patterns, the aliasing probability is independent of the MISR implementation and depends only upon the underlying polynomial. This is a generalization of the isomorphism identified for the single-input case by Serra et al. [17]. The CA and LFSR registers are isomorphic in the sense that they have equivalent cycle structures up to relabeling of the states under linear operation [17].

However, if the probability of occurrence is not the same for all error patterns, Miller's results show that the probability of aliasing is different for feed-forward and feedback LFSR implementations as well as for the HCA configurations which he considered. As such, it is reasonable also to conjecture that the probability of aliasing should be different for the two implementations of C-MISR circuits introduced in this paper.

In [44], an error model assigning a fixed probability of error p to each bit of the error pattern was suggested. In this manner, the probability of an error pattern is a function of the number of 1s in the pattern and hence of the Hamming distance between the error and the error-free words. For $p < 0.5$, patterns with fewer ones are more likely. This is arguably a more realistic error model than assuming that all error patterns are equally likely. Preliminary results from [44] indicate that for small p, the HCA-based MISRs display better aliasing characteristics than the corresponding LFSR-based MISRs. Although this was the error model used in our simulations, we were unable to distinguish this difference, perhaps in part due to the limited range of p we were considering.

Table 3 CA rules implementing a cyclic group for both null and cyclic boundary conditions. Cellular automaton 90h150 refers to a rule 90 and 150 hybrid cellular automaton where certain sites implement CA rules 90 or 150 along the array.

Rule	*Equation*	*Boundary*	*Length*
204	$a_i(t)$	all	all
51	$a_i(t)$	all	all
60	$a_{i+1}(t) \oplus a_i(t)$	null	all
195	$a_{i+1}(t) \oplus a_i(t)$	null	all
102	$a_i(t) \oplus a_{i-1}(t)$	null	all
153	$a_i(t) \oplus a_{i-1}(t)$	null	all
90	$a_{i+1}(t) \oplus a_{i-1}(t)$	null	4,6,8,$\cdots$
165	$a_{i+1}(t) \oplus a_{i-1}(t)$	null	4,6,8,$\cdots$
150	$a_{i+1}(t) \oplus a_i(t) \oplus a_{i-1}(t)$	null	4,6,8,$\cdots$
105	$a_{i+1}(t) \oplus a_i(t) \oplus a_{i-1}(t)$	null	4,6,8,$\cdots$
240	$a_{i+1}(t)$	cyclic	all
15	$a_{i+1}(t)$	cyclic	all
170	$a_{i-1}(t)$	cyclic	all
85	$a_{i-1}(t)$	cyclic	all
150	$a_{i+1}(t) \oplus a_i(t) \oplus a_{i-1}(t)$	cyclic	all
105	$a_{i+1}(t) \oplus a_i(t) \oplus a_{i-1}(t)$	cyclic	all
101	$[a_{i+1}(t) \cup a_i(t)] \oplus a_{i-1}(t)$	cyclic	5,7,9,$\cdots$
154	$[a_{i+1}(t) \cup a_i(t)] \oplus a_{i-1}(t)$	cyclic	5,7,9,$\cdots$
89	$[a_{i+1} \cup a_i(t)] \oplus a_{i-1}(t)$	cyclic	5,7,9,$\cdots$
166	$[a_{i+1} \cup a_i(t)] \oplus a_{i-1}(t)$	cyclic	5,7,9,$\cdots$
75	$a_{i+1}(t) \oplus [a_i(t) \cup a_{i-1}(t)]$	cyclic	5,7,9,$\cdots$
180	$a_{i+1}(t) \oplus [a_i(t) \cup a_{i-1}(t)]$	cyclic	5,7,9,$\cdots$
45	$a_{i+1}(t) \oplus [a_i(t) \cup a_{i-1}(t)]$	cyclic	5,7,9,$\cdots$
210	$a_{i+1}(t) \oplus [a_i(t) \cup a_{i-1}(t)]$	cyclic	5,7,9,$\cdots$
90h150	$a_{i+1}(t) \oplus a_{i-1}(t)$ $a_{i+1}(t) \oplus a_i(t) \oplus a_{i-1}(t)$	null	all

Summary

The main contribution of this work was improvement upon conventional test pattern generator circuitry for built-in self-test, thereby increasing the fault-detection capability of the apparatus and the means by which testability is achieved. The methodology is centered around the concept of employing one-dimensional cellular automata (CA) as alternatives to linear feedback shift registers (LFSRs). The use of these structures as primary "driving engines" driving logic arrays results in a WTPG function with improved statistical properties, wiring complexities, and performance.

In retrospect, it was determined that the CA-based WTPGs exhibited much better local and global random properties, and, with reservations, appeared similar to the statistically independent model. By virtue of the local communication architecture and regular topology of the CA-based WTPGs, there is a reduced wiring complexity associated with the development of such BIST test circuitry. [In this regard, they are extendible to scanning techniques such as boundary scan, and to observation techniques incorporating a weighted cellular automaton logic block observer (WCALBO) with multiple distributions.] In particular, it was learned that the rule 90/150 HCA, under the zero-spacing WTPG configuration, demonstrated sufficiently acceptable random properties and is well suited to VLSI.

Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada, the Province of Manitoba's Strategic Research Grant Program, and the Canadian Microelectronics Corporation. We would also like to acknowledge W. Pries for the maximal-length HCA table, and D. M. Miller, J. C. Muzio, and M. Serra, VLSI Design and Test Group, University of Victoria, for the early dissemination of their SA results.

References

1. T. W. Williams and K. P. Parker, "Design for Testability—A Survey," *Proc. IEEE* **71,** 98–112 (1983).
2. E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testing," *Proceedings of the 14th Design Automation Conference*, New Orleans, June 1977, pp. 462–468.
3. S. Funatsu, N. Wakatsuki, and T. Arima, "Test Generation Systems in Japan," *Proceedings of the 12th Design Automation Conference*, Boston, June 1975, pp. 114–122.
4. H. Ando, "Testing with Random-Access Scan," *Proceedings of Compcon 80*, San Francisco, February 1980, pp. 50–52.
5. J. H. Stewart, "Future Testing of Large LSI Circuit Cards," *Proceedings of the IEEE Semiconductor Test Symposium*, October 1977, pp. 6–17.
6. B. Konemann, J. Mucha, and G. Zwiehoff, "Built-In Logic Block Observation Techniques," *Proceedings of the 1979 IEEE International Test Conference*, Cherry Hill, NJ, October 1979, pp. 37–41.
7. S. W. Golomb, *Shift Register Sequences*, Holden-Day Publishing Co., San Francisco, 1982.
8. H. D. Schnurmann, E. Lindbloom, and R. G. Carpenter, "The Weighted Random Test-Pattern Generator," *IEEE Trans. Computers* **C-24,** 695–700 (July 1975).
9. J. Savir, G. S. Ditlow, and P. H. Bardell, "Random Pattern Testability," *IEEE Trans. Computers* **C-33,** 79–90 (January 1984).
10. C. K. Chin and E. J. McCluskey, "Weighted Pattern Generation for Built-In Self-Test," *Technical Report No. 84-7*, Center for Reliable Computing, Stanford University, Stanford, CA, 1984.
11. P. D. Hortensius, "Parallel Computation of Non-Deterministic Algorithms in VLSI," Ph.D. Dissertation, University of Manitoba, Winnipeg, Manitoba, Canada, 1987.
12. S. Wolfram, "Statistical Mechanisms of Cellular Automata," *Rev. Mod. Phys.* **55,** 601–644 (1983).
13. P. H. Bardell and W. H. McAnney, "Parallel Pseudorandom Sequences for Built-In Test," *Proceedings of the 1984 IEEE International Test Conference*, pp. 302–308.
14. P. D. Hortensius, R. D. McLeod, W. Pries, D. M. Miller, and H. C. Card, "Cellular Automata-Based Pseudorandom Number Generators for Built-In Self-Test," *IEEE Trans. Computer-Aided Design* **8,** 842–859 (August 1989).
15. S. Wolfram, "Random Sequence Generation by Cellular Automata," *Adv. Appl. Math.* **7,** 127–169 (1986).
16. W. Pries, A. Thanailakis, and H. C. Card, "Group Properties of Cellular Automata and VLSI Applications," *IEEE Trans. Computers* **C-35,** 1013–1024 (December 1986).
17. M. Serra, D. M. Miller, and J. C. Muzio, "Linear Cellular Automata and LFSRs Are Isomorphic," *Proceedings of the Third Technical Workshop, New Directions for IC Testing*, Halifax, Nova Scotia, Canada, October 26–27, 1988, pp. 195–205.
18. S. Zhang, D. M. Miller, and J. C. Muzio, "The Determination of Minimum Cost One-Dimensional Linear Cellular Automata with Maximum Length Cycles," submitted to *Electron. Lett.* (1989).
19. B. W. Podaima, "Weighted Pattern Generation for Built-In Self-Test Using Cellular Automata," M.Sc. Thesis, University of Manitoba, Winnipeg, Manitoba, Canada, 1989.
20. H.-J. Wunderlich, "Multiple Distributions for Biased Random Test Patterns," *Proceedings of the 1988 IEEE International Test Conference*, pp. 236–244.
21. J. A. Waicukauski and E. Lindbloom, "Fault Detection Effectiveness of Weighted Random Patterns," *Proceedings of the 1988 IEEE International Test Conference*, pp. 245–250.
22. F. Brglez, P. Pownall, and R. Hum, "Accelerated TPG and Fault Grading Via Testability Analysis," *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, Kyoto, Japan, June 1985, pp. 695–698.
23. F. Brglez, C. Gloster, and G. Kedem, "Hardware-Based Weighted Random Pattern Generation for Boundary Scan," *Technical Report TR89-11*, Microelectronics Center of the University of North Carolina, Research Triangle Park, 1989.
24. K. S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.
25. D. Knuth, *Seminumerical Algorithms*, Addison-Wesley Publishing Co., Reading, MA, 1981.
26. W. H. Press, B. P. Flannery, and W. T. Vetterling, *Numerical Recipes—The Art of Scientific Computing*, Cambridge University Press, New York, 1986, pp. 381–396.
27. A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill Book Co., Inc., New York, 1982.
28. C. Gloster and F. Brglez, "Boundary Scan with Built-In Self Test," *IEEE Design & Test of Computers* **6,** 36–44 (Feb. 1989).
29. E. Prange, "Cyclic Error-Correcting Codes in Two Symbols," *AFCRC-TN-57*, Vol. 103, Air Force Cambridge Research Center, Cambridge, MA, September 1957.
30. J. E. Meggitt, "Error Correcting Codes and Their Implementation," *IRE Trans. Info. Theory* **IT-17,** 232–244 (October 1961).
31. S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1983.
32. W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*, The MIT Press, Cambridge, MA, 1972.
33. H. Fujiwara, *Logic Testing and Design for Testability*, The MIT Press, Cambridge, MA, 1985.
34. P. H. Bardell, W. H. McAnney, and J. Savir, *Built-In Self Test for VLSI: Pseudorandom Techniques*, John Wiley & Sons, Inc., New York, 1987.
35. R. A. Frohwerk, "Signature Analysis: A New Digital Field Service Method," *Hewlett Packard J.*, pp. 2–8 (May 1977).
36. J. E. Smith, "Measures of the Effectiveness of Fault Signature Analysis," *IEEE Trans. Computers* **C-29,** 510–514 (June 1980).
37. J. C. Muzio, F. Ruskey, R. C. Aitken, and M. Serra, "Aliasing Probabilities of Some Data Compression Techniques," *Developments in Integrated Circuit Testing*, D. M. Miller, Ed., Academic Press, Inc., New York, 1987.
38. T. W. Williams, C. W. Starke, W. Daehn, and M. Gruetzner, "Comparison of Aliasing Probabilities for Primitive and Non-Primitive Polynomials," *Proceedings of the 1986 IEEE International Test Conference*, pp. 282–288.
39. N. Benowitz, D. F. Calhoun, G. E. Alderson, J. E. Baver, and C. T. Joeckel, "An Advanced Fault Isolation System for Digital Logic," *IEEE Trans. Computers* **C-24,** 489–497 (May 1975).
40. S. Z. Hassan, D. J. Lu, and E. J. McCluskey, "Parallel Signature Analyzers—Detection Capabilities and Extensions," *Proceedings of COMPCON 83*, 1983, pp. 440–445.
41. R. David, "Signature Analysis for Multiple-Output Circuits," *IEEE Trans. Computers* **C-35,** 830–837 (1986).
42. P. D. Hortensius, R. D. McLeod, and H. C. Card, "Cellular Automata-Based Signature Analysis for Built-In Self-Test," *IEEE Trans. Computers*, in press.
43. J. L. Carter, "The Theory of Signature Testing for VLSI," *Proceedings of the 14th Annual ACM Symposium on the Theory of Computing*, 1982, pp. 66–76.
44. D. M. Miller and S. Zhang, "Aliasing in Multiple-Input Data Compactors," presented at the Canadian Conference on Electrical and Computer Engineering, Montreal, Quebec, Canada, September 17–20, 1989.

Received May 16, 1989; accepted for publication September 20, 1989

Peter D. Hortensius *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.*
Dr. Hortensius received his B.Sc.E.E., M.Sc., and Ph.D. degrees in 1982, 1985, and 1987, all from the University of Manitoba. He joined the IBM Thomas J. Watson Research Center as a Research Staff Member in 1987 and is currently a member of the Workstation Design Group. Dr. Hortensius' research interests include processor design, system architecture, portable computing systems, IC design and testing, and computer applications to aid the handicapped.

Robert D. McLeod *Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, Manitoba, Canada.*
Dr. McLeod received his Bachelor's degree in electrical engineering in 1981, his Master's degree in 1983, and his doctorate in 1985, all from the University of Manitoba. He is currently an assistant professor in electrical engineering at the University of Manitoba. Dr. McLeod's research interests include the physics of VLSI, fabrication, IC design, testing, and fault-tolerant computing.

Blake W. Podaima *Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, Manitoba, Canada.*
Mr. Podaima is currently a Ph.D. student in electrical engineering at the University of Manitoba. He received his B.Sc. and M.Sc., also from the University of Manitoba, in 1985 and 1989, respectively. Mr. Podaima's research interests include VLSI testing and statistical pattern recognition. He is currently the recipient of a National Science and Engineering Research Council Postgraduate Scholarship.