

On Hardware Generation of Random Single Input Change Test Sequences

R. David* P. Girard** C. Landrault** S. Pravossoudovitch** A. Virazel**

* LAG (INPG - CNRS - UJF), BP 46, 38402 St-Martin-d'Hères, France
Rene.David@inpg.fr

** LIRMM - CNRS / Université Montpellier II, 161 rue Ada, 34392 Montpellier, France
[name]@lirmm.fr

Abstract

The combination of higher quality requirements and sensitivity of high performance circuits to delay defects has led to an increasing emphasis on delay testing of VLSI circuits. As delay testing using external testers requires expensive ATE, built-in self test (BIST) is an alternative technique that can significantly reduce the test cost.

It has been proven that Single Input Change (SIC) test sequences are more effective than classical Multiple Input Change (MIC) test sequences when a high robust delay fault coverage is targeted. It has also been shown that random SIC (RSIC) test sequences achieve a higher fault coverage than random MIC (RMIC) test sequences when both robust and non robust tests are under consideration; the experimental results were based on a software generation of RSIC sequences that are easily generated.

Obviously, an hardware RSIC generation providing similar results can be obtained. However, this hardware generator must be carefully designed. In this paper, it is explained what are the criteria which must be satisfied for this purpose. A solution is proposed and illustrated with an example. Then, it is shown that a bad result may be obtained if one of these criteria is not satisfied. The next step of this study will be the estimation of the overhead implied by the proposed RSIC generator.

Keywords: BIST, delay testing, random testing, single input change, hardware generation.

1 Introduction

Built-In Self-Test (BIST) has been proposed as a powerful design for testability technique for addressing the highly complex VLSI testing problems [1]. BIST design includes on-chip circuitry to provide test patterns and to analyze output responses. It can perform the test internal to the chip so that the need for complex external testing equipment is greatly reduced. Using BIST, the test volume can be significantly reduced, and many of the traditional testing problems (low accessibility of internal nodes that increases the test complexity) can be overcome [2]. Another interesting feature of BIST strategies is that they allow at-speed testing of the circuit under test, i.e. test at the nominal operation frequency.

At-speed testing is becoming an essential part of the verification process of today's VLSI circuits since it allows to optimize the test time and provides the means to test for *delay faults*. A delay fault occurs in a circuit when one or more paths in the circuit fail to propagate a signal within the time interval specified by the clock period. Detection of delay faults requires *two-pattern tests*. An

initialization vector is applied and the circuit is allowed to stabilize. Then, the *test vector* is applied and the circuit outputs are sampled at clock speed. The response is then compared to that of the fault-free circuit to determine the presence or the absence of a delay fault.

Delay fault testing requires two pattern tests, i.e., input changes. SIC test sequences have been investigated in the literature. SIC test pairs are sufficient to detect all robustly detectable path delay faults [3], with a test length shorter than that required with MIC test pairs [4]. This fact has motivated the development of BIST techniques in which SIC test pairs are generated for testing delay faults [4][5]. It was also observed that a random SIC sequence is efficient when non-robust tests and their validation by other tests are considered [6].

The effectiveness of RSIC test sequences for delay faults was shown in [7], on the basis of simulation performed on the *combinational* parts of various circuits of the ISCAS'89 benchmark set [8]. In [7], it is shown that RSIC test sequences achieve a higher fault coverage than RMIC test sequences when both robust and non robust tests are under consideration. The RSIC and RMIC sequences are compared in the following way. For each case, i.e., for a RSIC sequence and for a RMIC sequence, the fault coverage is estimated as follows: 1) *all the faults* for which at least a robust test has been found are definitely tested by the random sequence under consideration; 2) *a percentage* (called *success rate*) of the other faults for which at least a non-robust test has been found is also tested by the random sequence under consideration. In [7], it was observed that, for any success rate, the fault coverage of the RSIC sequence becomes higher than the fault coverage of the RMIC sequence when the test length increases.

A RSIC sequence may be software or hardware generated. In [7], the experimental results were based on a software generation of RSIC sequences because it is very easy to generate. In the present paper, a hardware generation that allows to get the same results than those achieved by the software generation is presented. For this hardware generation to be as effective as the software generation, a number of criteria must be satisfied. These criteria are defined and discussed in the present paper. In particular, it is shown that, if some of the criteria is not satisfied by the hardware generation, the test result may be very bad. In [9], the author says that "*random numbers should not be generated with a method chosen at random*"; this paper illustrates that the same comment applies to random test sequences, particularly RSIC test sequences in our case.

In [7], two classes of faults defined from the tests derived from TestGen [10] were considered. The first kind of test is practically like a robust test. This is explained in [7], where such a test is called a pseudo-robust test (PR-test); the first class of faults corresponds to the set of faults such that at least one PR-test exists (these faults are *PR-testable*). The second class of faults corresponds to all the faults such that at least one test (robust or not) exists; they are said to be *NR-testable* (standing for non-robustly) and this set includes the set of PR-testable faults.

Using TestGen [10], a deterministic generator, all the PR-testable and NR-testable faults can be known. Hence, the effectiveness of a RSIC generator can be evaluated thanks to two basic measurements, the *PR-efficiency* (coverage of PR-testable faults by a PR-test), and the *NR-efficiency* (coverage of NR-testable faults).

The rest of the paper is organized as follows. Section 2 presents theoretical bases of an hardware generation of RSIC test sequences. In Section 3, it is illustrated that the proposed hardware generation provides similar results compared to a software generation. Section 4 shows that bad results can be obtained if the hardware generation is not correctly performed. In the concluding remarks (Section 5),

it is explained that some constraints of the generation in Section 3 may be relaxed without significant lost in test quality.

2 Theoretical bases of a generation of RSIC test sequences

In Section 2.1, the three criteria that have to be satisfied by a RSIC generator are presented. Then, a solution for each criterion is given in Sections 2.2 to 2.4.

2.1 Criteria to be satisfied

Let us first define what a RMIC and a RSIC sequence should be from a theoretical point of view (we assume implicitly the case of equal likelihood of all vectors). Let

$$S = V(1)V(2)...V(l)...V(L), \quad (1)$$

be a test sequence composed of L successive n -bit vectors $V(l)$. Each vector takes a value from the set

$$V = \{V_0, V_1, \dots, V_{2^n-1}\}, \quad (2)$$

where V_j corresponds to the n -bit vector associated with the decimal value j . For example, for $n = 5$, $V_9 = 01001$, i.e., $x_1 = x_3 = x_4 = 0$ and $x_2 = x_5 = 1$.

In a RMIC sequence, the probability $\Pr[V(l) = V_j] = 1 / 2^n$ for any l and any j , and the probability $\Pr[V(l) = V_j]$ is independent of the values $V(i)$, where $i = 1, 2, \dots, l-1$.

In a RSIC sequence,

$$\Pr[V(l) = V_j \mid V(l-1) = V_k] = \frac{1}{n}, \text{ if and only if } |j - k| = 2^a, \quad (3)$$

where a is a non-negative integer. In other words, for any $l > 0$, $V(l)$ differs from $V(l-1)$ by exactly *one bit randomly drawn*. In addition, this bit must be *independent of the bits drawn before*, which can be expressed by:

$$\Pr[V(l) = V_j \mid V(l-2) = V_j] = \frac{1}{n^2}, \quad (4)$$

i.e., the input variable changing at time l is independent of the input variable changing at time $(l-1)$.

Since any software or hardware generation of "random" numbers is in fact "pseudorandom", the generation of a RMIC or RSIC sequence cannot be "perfect". It is then important to be careful in order to achieve a "good" generation, with respect to the basic principle.

Figure 1 represents the principle of the generation of a RSIC test sequence. Basically a k -bit random (in fact pseudorandom) source is used. A software generation can be performed thanks to an instruction "random" drawing a random number uniformly distributed in the range $[0, 1)$; present day programming languages use a recurrent relation producing a sequence called *linear congruential* [9]. A hardware generation is usually based on a *maximal length LFSR*¹ (linear feedback shift register). The value of the vector $Q_1Q_2\dots Q_k$ changes at each time unit (clock cycle of the test session).

¹ A LFSR is maximal length if its characteristic polynomial has some mathematical property: it must be primitive. Such a LFSR generates a sequence of bits which is pseudorandom. The properties of these generators and some useful polynomials can be found in various books including [11].

At each time l , a subset of m bits is used ($m \leq k$). The vector $R(l) = R_1(l)R_2(l)\dots R_m(l)$ is transformed into a n -bit vector $V(l) = x_1(l)x_2(l)\dots x_n(l)$ which is applied to the circuit under test [11].

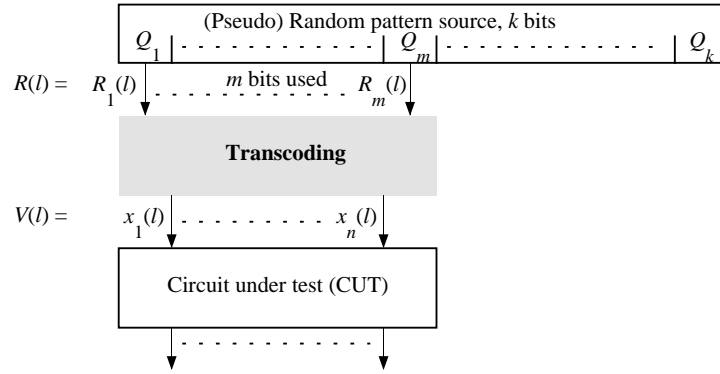


Fig. 1. Principle of generation of a RSIC test sequence

A RSIC test sequence corresponding to the previous definition can be obtained if and only if the three following criteria are satisfied.

Criteria for the generation of a RSIC test sequence:

Criterion 1: The transcoding between $R(l)$ and $V(l)$ satisfies Equ. (3).

Criterion 2: $R(l)$ is independent of $R(l-1)$.

Criterion 3: The period of the sequence $S = V(1)V(2)\dots V(l)\dots$ is greater than the test length L . □

If Criterion 1 is satisfied, then every input x_i has the same probability $1/n$ to change at time l , i.e., Equ. (3) is verified.

Given the vector $V(l-1)$, the input x_i changing at time l (i.e., such that $x_i(l) \neq x_i(l-1)$, while $x_j(l) = x_j(l-1)$ for every $i \neq j$), depends on the vector $R(l)$ via the transcoding shown in Fig. 1. If Criterion 2 is satisfied, then the condition in Equ. (4) is verified.

If the period P was shorter than L , then the subsequence $V(P+1)V(P+2)\dots$ would test exactly the same faults as the subsequence $V(1)V(2)\dots$ (because the circuit is combinational). Hence the fault coverage of the test sequence $S = V(1)V(2)\dots V(P)\dots V(L)$ would be the same as the fault coverage of its prefix $S_1 = V(1)V(2)\dots V(P)$. This is avoided if $P > L$.

Hence, if the three criteria are satisfied, the RSIC test sequence generated satisfies the theoretical constraints.

These constraints are relatively easy to satisfy by a software generation. Let us now specify how these criteria may be satisfied by a hardware generation. The contents of Sections 2.2 and 2.3 are given in [11]; they are recalled here in order to present a self-contained paper.

2.2 Transcoding from R to V

The principle is shown in Fig. 2. The vector $R(l)$ is mapped into a 1-out-of- n vector $T(l)$. Every component of $T(l)$ is applied to the input of a T flip-flop. Hence, given a random vector $R(l)$, it implies a random trigger input $T_i(l) = 1$ while other trigger inputs $T_j(l) = 0$ for $j \neq i$. Hence, $V(l)$ is similar to $V(l-1)$ excepted the value of x_i .

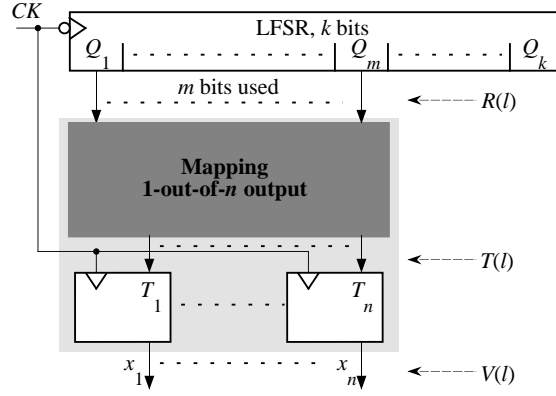


Fig. 2. Transcoding from $R(l)$ to $V(l)$.

If $n = 2^a$, where a is an integer, then $m = a$ can be chosen. However, the number k of bits in the LFSR must be greater than m in order that all the m -bit vectors have a non-zero probability.

If $2^{a-1} < n < 2^a$, then one must have $m \geq a$, i.e., $m \geq \lceil \log_2 n \rceil$. In practice,

$$m = \lceil \log_2 n \rceil + \alpha \quad (5)$$

will be chosen in order that all the variables x_i change with approximately the same probability. In Section 3, it will be observed that $\alpha = 2$ or 3 is certainly sufficient.

Assume, for example, that $n = 3$. Then $m \geq \lceil \log_2 3 \rceil = 2$. If $m = 3$ is chosen, there are $2^m = 8$ possible values for $R(l)$. Let us denote them R_0, R_1, \dots, R_7 . If among these values, 3 are associated with T_1 , 3 are associated with T_2 , and 2 are associated with T_3 , one may have: $T_1 = R_0 + R_1 + R_2$, $T_2 = R_3 + R_4 + R_5$, and $T_3 = R_6 + R_7$. It follows that $\Pr[x_1 \text{ changes}] \approx \Pr[x_2 \text{ changes}] \approx 3/8$, and $\Pr[x_3 \text{ changes}] \approx 2/8$.

Note that $\Pr[R_0]$, corresponding to $R = 0\dots 0$, is a bit lower than $\Pr[R_j]$ for all other values of j . This vector may be associated with one of the T_q gathering the greatest number of R_j (T_1 or T_2 in our example), for a better balance of the probabilities.

2.3 Multiple shifting between two successive vectors $R(l)$

A solution to satisfy this criterion is also given in [11]. Let us illustrate it with an example such that $k = 5$ and $m = 2$.

The polynomial $x^5 \oplus x^2 \oplus 1$ is primitive. The sequence of bits generated by a LFSR having this characteristic polynomial is pseudorandom and its period is $2^5 - 1 = 31$ as illustrated in Fig. 3. The successive bits in the pseudorandom sequence are named $b(1), \dots, b(31)$.

Vector $R(l)$ could be made of Q_1 and Q_2 (leftmost bits in the LFSR). As illustrated in Fig. 3.a, if the first vector is $R(1) = b(5)b(4) = 00$, then, after a single shifting, $R(2) = b(6)b(5) = 10$. Hence, for any time l , $R_2(l+1) = R_1(l)$, i.e., Criterion 2 is not satisfied since $R(l+1)$ is not independent of $R(l)$.

If $\sigma = 2$ shiftings are performed between $R(l)$ and $R(l+1)$, then there is not any bit $b(i)$ which can be found in both $R(l)$ and $R(l+1)$. See Fig. 3.b. Since $\sigma = 2$ and the period $2^5 - 1 = 31$ do not share a common factor, the period obtained for the successive vectors $R(l)$ is also 31.

In [11], it is shown that σ shiftings (2 shiftings in our example) can be obtained in a single clock pulse thanks to a modification of the feedback of the LFSR. The only conditions to be verified are:

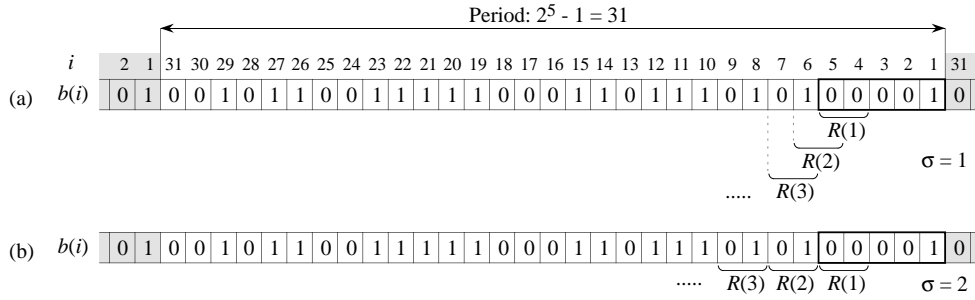


Fig. 3. Example with $k = 5$ and $m = 2$. (a) Shifting $\sigma = 1$. (b) Shifting $\sigma = m = 2$.

$$m \leq \sigma \leq (2^k - 1) - m \text{ (furthermore } \sigma = m \text{ is recommended);} \quad (6)$$

$$\sigma \text{ and } 2^k - 1 \text{ do not share a common factor.} \quad (7)$$

Condition (6) is required for $R(l)$ and $R(l+1)$ being independent, and Condition (7) for the sequence $R(1)R(2)\dots$ having the maximal period $2^k - 1$.

2.4 Period of the sequence generated

The period of the sequence of bits generated by an LFSR of length k whose polynomial is primitive is $2^k - 1$. If, at every time l , we consider a vector of successive bits in the register (as in Fig. 3.a) the period of the sequence of vectors is also $2^k - 1$. If there are σ shiftings between two successive vectors (as in Fig. 3.b), the period remains $2^k - 1$ if and only if σ and $2^k - 1$ do not share a common factor. This last scheme corresponds to the generation of an RMIC sequence. Now, what about the period of a RSIC sequence generated as explained in Section 2.2? The answer is given by Theorem 1 below. Let us first specify what a H-generator (standing for hardware generator) is.

H-generator 1. This hardware RSIC generator corresponds to the scheme in Fig. 1, with the transcoding explained in Section 2.2, such that.

- 1) The LFSR is maximal-length (period $2^k - 1$), and $2 \cdot (2^k - 1) > L$.
- 2) $m \geq \lceil \log_2 n \rceil$.
- 3) The number σ of shiftings (Section 2.3) is such that: $1 \leq \sigma < 2^k - 1$, and σ does not share a common factor with $2^k - 1$.
- 4) Let b_j be the number of m -bit vectors associated with T_j ; $\max_i b_j - \min_i b_j \leq 1$.

□

Condition 2 in H-generator 1 allows all the inputs having a non-zero probability of changing. Conditions 4, 3, and 1 allow to satisfy criteria 1 (approximately), 2, and 3, respectively.

According to Section 2.2, the 2^m vectors $R_0, R_1, \dots, R_{2^m-1}$ are associated with the trigger inputs T_1 to T_n . The number of vectors R_i associated with T_j is denoted by b_j . For example, if $T_1 = R_0 + R_1 + R_2$, then $b_1 = 3$. Let us remind that R_0 , corresponding to $R = 0\dots 0$, appears a bit less often than other R_i 's.

Theorem 1. If the vector R_0 is associated with the trigger input T_j and there is T_k such that $b_j = b_k$, then the period of the RSIC sequence obtained from H-generator 1 is

$$2 \times (2^k - 1). \quad (8) \quad \square$$

The proof is given in Appendix. The condition $b_j = b_k$ in the theorem is certainly true if $n = 2^m$, and can always be satisfied if $n \neq 2^m$.

From Theorem 1, it follows that, given L , k must be chosen such that $2 \times (2^k - 1) > L$; i.e.,

$$k > \log_2 \left(\frac{L}{2} - 1 \right) \approx \log_2 \frac{L}{2} \quad (9)$$

In summary, the hardware RSIC generation in this section will take into account Equations (6), (7), (5), and (9) for the choice of parameters m , σ , and k . In addition, it is necessary that $k \geq m$ (Fig. 1) but this condition is generally satisfied when (9) is verified.

We also suggest to use 3-term polynomials for the LFSR. As a matter of fact, the number of 2-input XOR gates necessary for σ shiftings is σ for a 3-term polynomial, while it should be $3 \times \sigma$ for a 5-term polynomial (primitive polynomials with an even number of terms do not exist).

3 Comparison of hardware and software generations

All the results in this section correspond to a case study: the combinational part of s382 circuit [8]. This is not a benchmark (we know that the results must be similar for hardware and software generations since the criteria defined in Section 2 are satisfied in both cases) but simply an illustration. For this circuit, we have $n = 24$. According to Equ. (5), $m \geq 5$ and, with $\alpha = 2$, $m = 7$ can be chosen, then from (6) $\sigma = 7$ can be chosen. There are 400 paths in the s382 circuit, hence 800 potential delay faults; among them, 734 are NR-testable out of which 704 are PR-testable.

We intend to simulate experiments up to $L = 1\,000\,000$. Hence from (9), $k \geq 19$ is required. We must be careful with the relation between σ and $2^k - 1$ (see Equ. (7)) and choose a 3-term polynomial (according to the end of Section 2.3). The smaller value of k greater than or equal to 19, such that there is a 3-term primitive polynomial, and such that no value lower than k has a common factor with $2^k - 1$ (i.e., any σ can be used) is 23 (Fig. 10.4 in [11]). The corresponding polynomial is $x^{23} \oplus x^5 \oplus 1$, or the dual one $x^{23} \oplus x^{18} \oplus 1$.

H-generator 2. This hardware RSIC generator is a particular case of H-generator 1 in which:

- 1) The polynomial of the LFSR is $x^{23} \oplus x^5 \oplus 1$.
- 2) $m = 7$.
- 3) The number of shiftings is $\sigma = 7$.
- 4) The number of m -bit vectors associated with T_i is $b_i = 6$ for $i = 1, \dots, 8$, and $b_i = 5$ for $i = 9, \dots, 24$.

□

Now, the performance of H-generator 2 will be compared with the performance of the software generator used in [7] and called S-generator here. This S-generator is based on an instruction called **rand** drawing a (pseudo)random number in the range $[0, 1)$.

S-generator. For a L -vector random SIC test sequence.

Step 1. Initialization: choose (deterministically or randomly) an initial test vector (n bits)

$$V(1) = x_1(1)x_2(1)\dots x_i(1)\dots x_n(1).$$

Step 2.

for $l = 2$ to L

```

Y = rand
i = [Y · n]
if  $x_i(l - 1) = 0$  then  $x_i(l) := 1$  else  $x_i(l) := 0$  end
for  $j \neq i$ ,  $x_j(l) := x_j(l - 1)$  end
V(l) :=  $x_1(l)x_2(l)\dots x_i(l)\dots x_n(l)$ 
end

```

3.1 Comparison of the PR-efficiencies of H-generator 2 and of S-generator.

The comparison is based on 15 experiments made with each generator. Each experiment corresponds to a generation with a different seed. The results obtained with the 15 random test sequences generated by the S-generator (named S1 to S15), and 15 test sequences generated by H-generator 2 (named S16 to S30) are presented in Fig. 4. The test length is $L = 1,000,000$ for every test sequence. The test length noted $L_D = 1398$ corresponds to the deterministic test sequence providing a 100 % efficiency for both PR-testable and NR-testable faults [10].

s382		PR-Efficiency (S-generator)														
#Vectors	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	Average
10	1.27	2.13	2.41	1.27	1.99	1.56	2.41	1.99	3.13	2.41	2.56	1.84	3.69	2.41	0.42	2.10
100	19.60	12.65	13.35	9.94	9.24	6.82	14.35	12.36	14.64	10.51	13.64	14.20	17.76	14.92	11.80	13.05
Ld/4	349	29.11	22.16	23.01	23.72	24.72	25.28	25.70	28.98	37.36	24.57	26.84	28.83	32.24	27.56	27.75
Ld/2	699	34.80	29.68	37.36	30.82	32.95	36.51	37.78	43.61	44.47	32.95	38.50	35.65	42.61	45.32	38.02
Ld	1398	48.44	47.31	52.84	47.44	44.32	48.16	50.85	52.70	53.98	45.17	49.58	53.98	55.68	55.68	50.63
2Ld	2796	60.66	65.20	66.76	59.81	60.80	62.36	61.93	62.65	61.80	57.53	61.23	65.20	65.20	70.60	62.95
5Ld	6990	74.72	80.40	77.27	73.86	75.14	73.86	75.14	75.99	77.27	71.59	77.41	77.13	74.43	77.98	75.85
10Ld	13980	80.25	86.08	85.65	81.39	81.39	81.67	82.52	82.52	82.39	82.81	83.81	82.39	81.95	83.52	82.86
20Ld	27960	84.66	87.78	87.35	86.93	87.07	86.93	85.65	85.23	86.22	86.65	87.22	87.64	86.08	86.65	86.62
50Ld	69900	87.78	89.49	89.91	88.92	88.92	88.49	88.64	88.07	88.35	88.77	89.20	89.20	88.92	88.92	88.82
100Ld	139800	88.64	90.06	90.63	88.92	90.06	89.06	89.34	88.64	88.77	89.49	90.76	90.19	89.77	89.49	89.52
1000000	88.92	90.06	90.63	89.20	90.06	89.34	89.49	88.92	88.92	89.91	90.76	90.19	89.91	89.49	89.06	89.66

s382		PR-Efficiency (H-generator 2)														
#Vectors	S16	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26	S27	S28	S29	S30	Average
10	1.14	3.13	2.13	2.27	3.13	1.99	2.56	1.42	1.42	2.69	1.56	1.84	1.99	1.84	2.27	2.09
100	4.55	18.32	14.20	18.32	16.63	5.82	6.53	4.83	5.53	6.53	5.68	5.68	5.82	5.82	5.68	6.66
Ld/4	349	27.98	26.70	25.14	28.98	26.84	27.41	26.70	26.99	25.99	30.40	28.69	26.27	28.41	22.58	27.11
Ld/2	699	36.22	38.78	37.78	43.61	42.61	36.22	37.50	35.36	36.36	38.22	39.49	36.22	37.22	32.81	37.64
Ld	1398	45.32	53.84	49.15	56.11	54.41	50.28	48.86	45.17	46.88	50.72	51.28	47.16	46.31	48.30	49.70
2Ld	2796	58.67	63.35	60.23	64.92	63.92	64.35	59.09	57.25	62.08	57.95	61.08	65.63	62.36	59.94	61.62
5Ld	6990	72.86	74.57	75.57	77.69	75.42	75.70	74.15	75.99	74.72	72.59	75.28	76.27	75.85	75.70	75.19
10Ld	13980	81.10	81.10	83.81	86.78	81.95	82.24	81.25	82.24	82.10	83.38	80.97	82.24	82.24	81.53	82.27
20Ld	27960	85.51	85.36	87.35	89.20	85.08	86.93	84.51	85.36	85.65	86.65	86.93	86.36	88.49	86.93	86.38
50Ld	69900	89.77	88.92	89.77	90.19	87.92	89.91	87.64	89.63	88.07	89.91	89.49	87.92	90.63	88.77	89.18
100Ld	139800	90.63	89.34	90.76	90.91	88.77	90.63	88.20	90.63	89.49	90.76	90.19	90.06	90.63	88.92	89.98
1000000	90.63	89.34	90.76	90.91	89.34	91.05	88.77	90.63	89.49	90.76	90.48	90.48	90.91	89.49	90.06	90.21

Fig. 4. PR-efficiency. (a) S-generator. (b) H-generator 2.

The results in Fig. 4 are illustrated by the graphs in Fig. 5, where the PR-efficiency is represented as a function of the test length for all the test sequences. For large test lengths, a difference between the results of the S-generator and H-generator 2 is not apparent. A detailed analysis of the results for large test lengths (namely $L = 139,800$ and $L = 1,000,000$) is shown in Fig. 6.

Figure 6 must be understood as follows. How many test sequences (among the 15 experimented) contain a PR-test for a proportion between 88 % and 89 % of the PR-testable faults if the test length is

$L = 139,800$? The answer is 4 for the software generation (among S1 to S15), and 3 for the hardware generation (among S16 to S30). For this range, the number of PR-testable fault for which no PR-test has been found is between 78 and 84. If the test length is $L = 1,000,000$, the number of sequences corresponding to the same coverages is 3 for software generation and 1 for hardware generation.

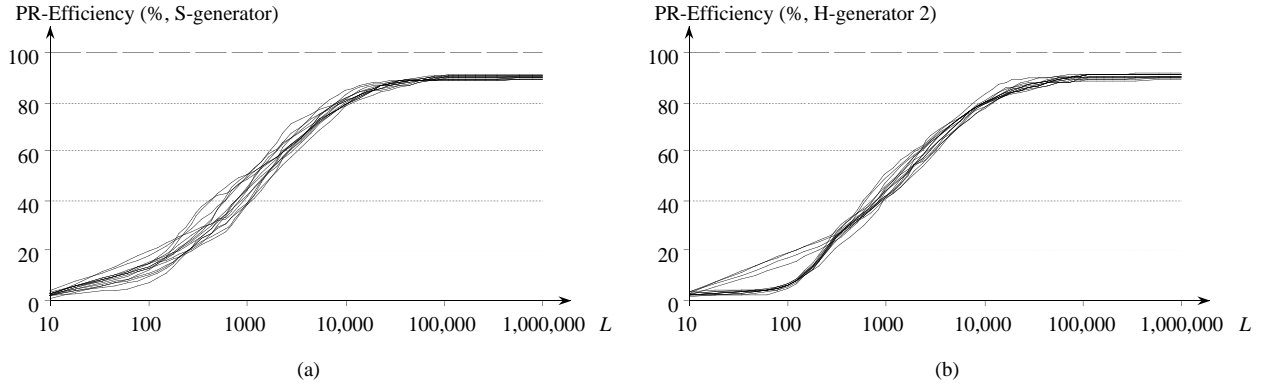


Fig. 5. Illustration of the results shown in Fig. 4. (a) S-generator. (b) H-generator 2.

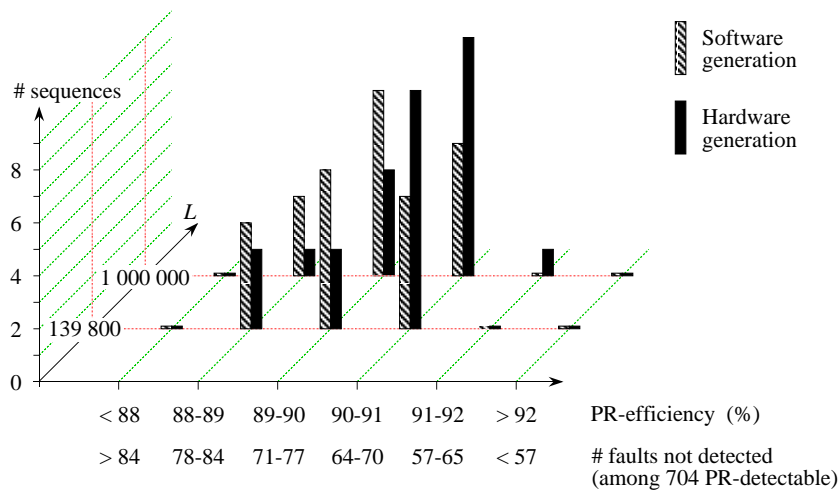


Fig. 6. Comparison of S-generator and H-generator 2 for large values of the test length.

The PR-efficiency is between 88 % and 91 % for all the test sequences (software and hardware) for $L = 139,800$. For $L = 1,000,000$, this efficiency is between 88 % and 91 % for the software generated sequences and between 88 % and 92 % for the hardware generated sequences.

In the results presented, more H-generated than S-generated sequences reach a high coverage. However, the difference is not significant: the results are quasi-identical.

3.2 Comparison of the NR-efficiencies of H-generator 2 and of S-generator.

Similarly to Figures 4 to 6, Figures 7 to 9 present the results for the NR-efficiencies.

The NR-efficiency is between 99.3 % and 100 % for all the test sequences (software and hardware) for $L = 139,800$. It is between 99.7 % and 100 % for all the test sequences (software and hardware) for $L = 1,000,000$.

Hence, from Sections 3.1 and 3.2, we can verify that the H-generator 2 and the S-generator produce practically the same results.

s382		NR-efficiency (S-generator)															
#Vectors	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	Average	
10	1.09	3.40	2.31	2.58	3.27	1.91	2.58	1.36	1.36	2.86	1.63	1.77	1.91	2.18	2.31	2.17	
100	4.49	19.89	14.99	18.80	17.71	5.72	6.67	4.63	5.99	6.54	5.85	5.99	5.58	6.13	5.58	8.97	
Ld/4	349	29.01	28.88	26.70	30.38	28.74	29.29	29.43	28.20	26.83	32.83	30.79	27.65	29.97	24.25	29.56	28.83
Ld/2	699	39.11	41.69	39.92	47.28	45.92	37.87	41.42	37.60	38.29	41.69	42.92	37.47	39.24	35.29	38.42	40.28
Ld	1398	48.37	58.72	52.73	60.49	58.18	54.22	53.55	48.23	50.14	55.18	56.40	50.82	49.46	53.13	56.27	53.73
2Ld	2796	63.08	69.08	64.85	70.57	69.21	69.48	64.58	62.27	68.12	63.22	67.03	71.79	67.72	66.08	69.62	67.11
5Ld	6990	79.69	82.56	83.11	85.01	83.38	82.69	84.74	80.93	84.05	81.47	79.69	82.96	83.92	83.51	83.65	82.76
10Ld	13980	89.65	90.46	91.96	95.37	90.74	90.46	92.10	90.74	91.41	91.68	90.46	91.28	91.14	90.05	90.19	91.18
20Ld	27960	94.82	95.37	96.59	98.22	95.37	95.77	95.77	94.68	95.64	95.50	96.32	95.77	97.55	96.86	94.82	95.94
50Ld	69900	99.04	99.46	99.04	99.31	98.64	98.91	98.91	99.04	98.50	99.18	99.04	97.55	99.59	99.18	99.18	98.97
100Ld	139800	99.86	99.86	100	100	99.46	99.59	99.46	100	100	100	99.73	99.59	99.59	99.31	99.73	99.74
1000000	99.86	99.86	100	100	100	100	100	100	100	100	100	100	100	100	99.86	100	99.97

s382		NR-efficiency (H-generator 2)															
#Vectors	S16	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26	S27	S28	S29	S30	Average	
10	1.22	2.18	2.58	1.22	2.04	1.49	2.58	2.18	3.54	2.31	2.86	1.77	4.49	2.45	0.40	2.22	
100	22.47	12.81	13.62	10.22	9.26	7.22	16.08	12.81	15.13	11.17	14.58	14.58	20.16	15.67	12.81	13.91	
Ld/4	349	33.65	22.75	23.43	24.65	26.02	26.56	28.47	31.19	40.60	26.29	29.01	29.97	35.84	29.70	39.24	29.82
Ld/2	699	39.65	31.06	39.65	32.70	34.60	38.83	42.10	46.74	47.96	35.69	41.56	38.15	47.14	48.92	51.77	41.10
Ld	1398	53.95	49.73	57.09	52.19	46.59	51.64	56.27	58.04	59.00	48.92	53.41	58.04	61.58	60.36	57.91	54.98
2Ld	2796	67.57	70.30	72.48	65.94	65.54	67.99	67.99	70.03	67.99	62.67	66.08	70.43	71.93	77.38	67.72	68.80
5Ld	6990	84.33	88.28	84.74	81.74	82.96	82.56	83.92	85.56	86.10	80.11	84.87	84.60	82.02	86.51	84.20	84.17
10Ld	13980	90.74	95.23	94.68	90.74	90.19	91.28	93.05	93.05	91.96	92.10	92.37	90.87	91.28	93.19	94.28	92.33
20Ld	27960	95.64	97.55	96.59	97.55	96.86	97.13	96.04	95.91	97.00	96.32	96.46	97.28	96.19	96.73	97.82	96.74
50Ld	69900	98.77	99.31	99.18	99.73	98.77	99.18	99.18	99.04	99.46	98.77	98.50	98.91	99.04	99.46	99.59	99.13
100Ld	139800	99.73	99.86	100	99.73	99.86	99.73	99.86	99.59	99.86	99.46	100	99.86	99.86	100	99.86	99.82
1000000	100	99.86	100	100	99.86	100	100	100	100	99.86	100	99.86	100	100	99.86	100	99.95

Fig. 7. NR-efficiency. (a) S-generator. (b) H-generator 2.

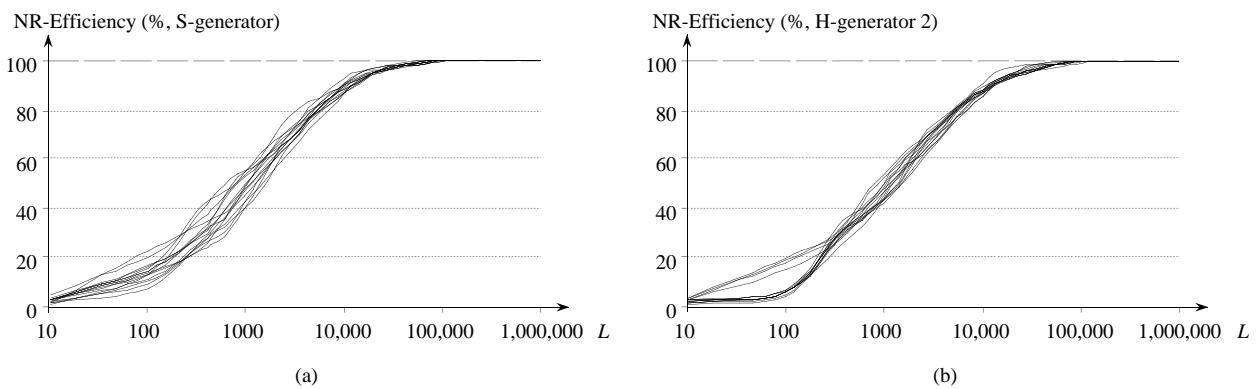


Fig. 8. Illustration of the results in Fig. 7. (a) S-generator. (b) H-generator 2.

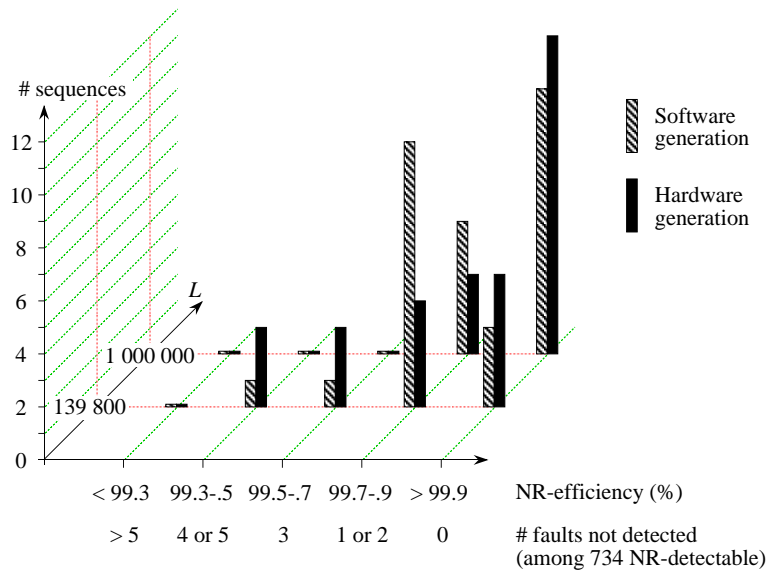


Fig. 9. Comparison of S-generator and H-generator 2 for large values of test length.

4 What happens if the hardware generation is not pertinent ?

In this section, it is shown that the efficiency of an hardware generated test sequence may be too low if the conditions in H-generator 1 are not satisfied, i.e., if the criteria in Section 2 are not satisfied. Let us consider successively the conditions 1 (criterion 3), 4 (criterion 1), and 3 (criterion 2) presented in H-generator 1 (it is quite obvious that a bad result would be obtained if condition 2 is not satisfied since at least one input of the circuit would never change). For the circuit under consideration s382, H-generator 2 is a specific mapping of the generic H-generator 1. Hence, the results obtained with a "bad" generation will be compared with the results obtained with the H-generator 2. For each case studied, a single test sequence is considered

4.1 The period is not enough long

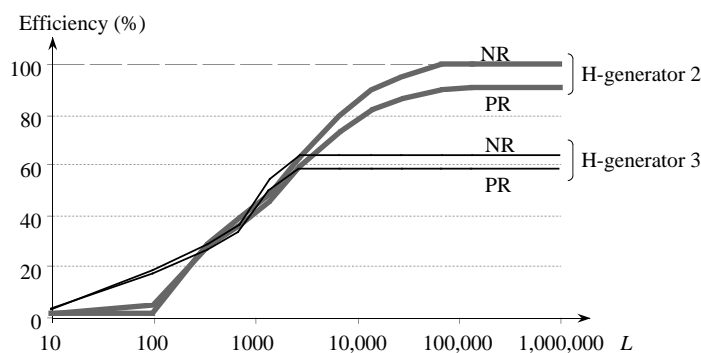


Fig. 10. Comparison of H-generator 3 (too short) with H-generator 2.

H-generator 3. Similar to H-generator 2, excepted that the polynomial of the LFSR is $x^{11} \oplus x^2 \oplus 1$ instead of $x^{23} \oplus x^5 \oplus 1$.

□

The polynomial $x^{11} \oplus x^2 \oplus 1$ has good properties: it is primitive and the period $(2^{11} - 1)$ does not share a common factor with the number of shiftings $\sigma = 7$. However it is too short. According to Theorem 1, the period of the RSIC sequence generated is $2 \times (2^{11} - 1) = 4094$. Accordingly, the test sequence is inefficient after the 4094th vector has been applied, as illustrated in Fig. 10.

4.2 Unequal changing probabilities

Condition 4 in H-generator 1 proposes that all inputs have approximately the same probability of changing at each new test vector. H-generator 4 does not fulfill this condition.

H-generator 4.

- 1) The polynomial of the LFSR is $x^{23} \oplus x^5 \oplus 1$.
- 2) $m = 14$.
- 3) The number of shiftings is $\sigma = 14$.
- 4) The number of m -bit vectors associated with T_i are $b_i = 1365$ for $i = 1, \dots, 4$, $b_i = 1364$ for $i = 5, \dots, 12$, and $b_i = 1$ for $i = 13, \dots, 24$.

□

From H-generator 4, the probability of changing of inputs x_{13} to x_{24} is very low in comparison with the probabilities of changings of x_1 to x_{12} . Figure 11 illustrates that this generator is not so good as H-generator 2.

Let us notice that, even if the phenomenon exists, it is not important quantitatively. For H-generator 4, we have the ratio ρ :

$$\rho = \frac{\text{Maximal changing probability}}{\text{Minimal changing probability}} = 1365.$$

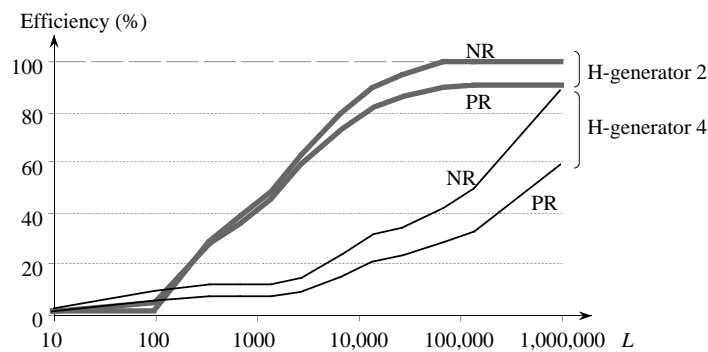


Fig. 11. Influence of unequal changing probabilities (H-generator 4).

If the ratio ρ is large enough (H-generator 4), the efficiency of the generator may not be good. But if this ratio is small, the difference with a generator perfect from this point of view (i.e., $\rho = 1$) is not important. We have observed that the ratio $\rho = 2$ does not affect significantly the results.

Hence, from now, the minimum possible value of m , i.e., $m = \lceil \log_2 n \rceil$ according to Equ. (5), will be used. If $n = 2^q$, where q is an integer, then $\rho = 1$. Otherwise $\rho = 2$.

4.3 The number σ of shiftings is too small

If the number σ of shiftings and the period of the sequence generated by the LFSR share a common factor, then the period of the RSIC sequence is obviously shortened. This is out of our hypothesis in this section. We only want to see what may happen if σ is too small, particularly if $\sigma = 1$.

Let us now define H-generator 5. This generator takes into account the result in Section 4.2, i.e. the value m is minimal. It is defined for various circuits, i.e. the value n is a parameter depending on the circuit. For the s382 circuit, the difference with H-generator 2 is that $m = \sigma = 5$ instead of 7 (hence $b_i = 1$ or 2).

H-generator 5.

- 1) The polynomial of the LFSR is $x^{23} \oplus x^5 \oplus 1$.
- 2) $m = \lceil \log_2 n \rceil$
- 3) The number of shiftings is $\sigma = m$.
- 4) The number of m -bit vectors associated with T_i is $b_i = 1$ or 2.

H-generator 6. Similar to H-generator 5, except that $\sigma = 1$.

□

H-generators 5 and 6 have been used for the combinational part of various circuits of the ISCAS 89 benchmark set. For every circuit, the applied RSIC test length was 100 times the corresponding deterministic test length obtained for a 100 % efficiency [10][7]. Here are the various numbers of inputs and random test lengths. Circuit s298: $n = 17$, $L = 68\,800$; s382: $n = 24$, $L = 139\,800$; s386: $n = 13$, $L = 73\,600$; s420: $n = 35$, $L = 132\,200$; s510: $n = 25$, $L = 136\,600$; s526: $n = 24$, $L = 139\,600$; s641: $n = 54$, $L = 277\,800$; s713: $n = 54$, $L = 441\,800$; s1238: $n = 32$, $L = 499\,200$; s1494: $n = 14$, $L = 351\,800$. The experimental results are presented in Fig. 12 for PR-efficiency and in Fig. 13 for NR-efficiency.

From Fig. 12 and 13, taking into account that a single experiment was performed for each case, one can observe the two following points. 1) The results of H-generators 5 are close to the results of the corresponding S-generators (this is a confirmation of the results in Section 3 and of the observation in Section 4.2 leading to the choice of minimal value of m). 2) The shifting $\sigma = 1$ (instead of the recommended $\sigma = m$) provides good results for many cases; however the results with $\sigma = 1$ are significantly worse than with $\sigma = m$ for two circuits (s386 and s1494).

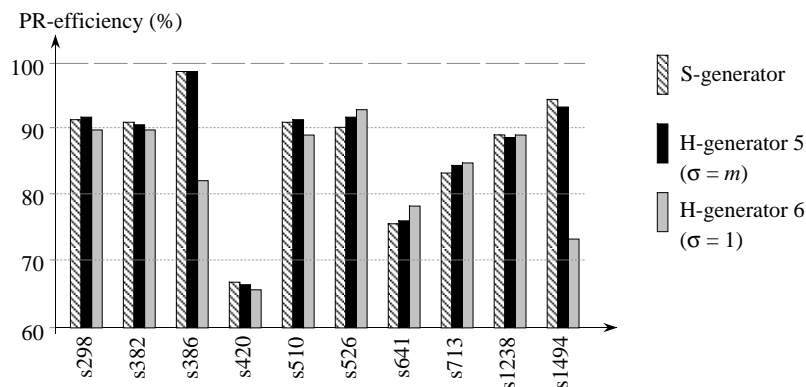


Fig. 12. Comparison of S-generator, H-generator 5, and H-generator 6 (PR-efficiency).

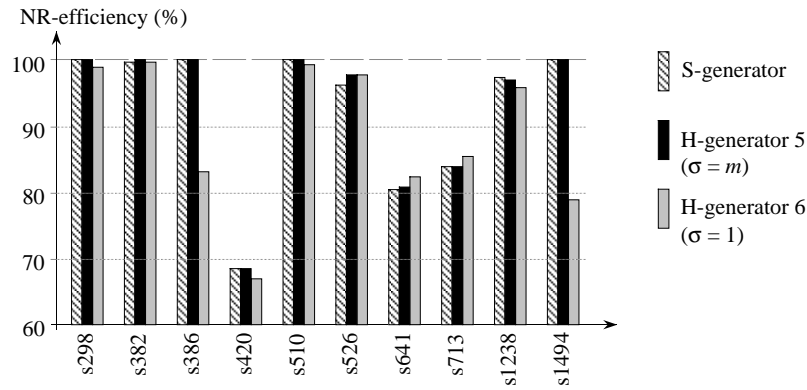


Fig. 13. Comparison of S-generator, H-generator 5, and H-generator 6 (NR-efficiency).

Let us analyse the reason of the influence of σ . Let V_a be any input vector, and V_b and V_c two vectors adjacent to V_a . If $\sigma = m$, we have:

$$\Pr[V(t+1) = V_b \mid V(t) = V_a] = 1 / n, \text{ and} \quad (10)$$

$$\Pr[V(t+1) = V_b \mid V(t) = V_a \text{ AND } V(t-1) = V_c] = 1 / n. \quad (11)$$

If $\sigma = 1$, Equ.(10) remains true while (11) is replaced by

$$\Pr[V(t+1) = V_b \mid V(t) = V_a \text{ AND } V(t-1) = V_c] = 1 / 2 \text{ for two vectors } V_b \text{ including } V_c. \quad (12)$$

$$\Pr[V(t+1) = V_b \mid V(t) = V_a \text{ AND } V(t-1) = V_c] = 0 \text{ for all the other vectors } V_b. \quad (12')$$

In the second case, there is a correlation between successive vectors. If a transition $V_c \rightarrow V_a$ occurs between $t-1$ and t , from (12) and (10) the probability that the same transition occurs between $t+1$ and $t+2$ is $1/4$ if $\sigma = 1$, while it is only $1/n^2$ if $\sigma = m$ (consistent with Equ. (4)). The effect of this correlation on the efficiency of the RSIC test sequence is difficult to predict. The results in Fig. 12 and 12 show that it cannot always be neglected.

5 Conclusion

In [7] the performance of a random SIC test sequence for delay testing was analyzed. The RSIC sequences used were software generated. In this paper, the criteria for a "good" RSIC sequence generation are presented. If they are respected (software or hardware generation) a good test result is obtained. However, one must be careful about the hardware generation: the effects of the various criteria on the efficiency of the test sequence are experimentally shown.

The next step of this study will be the estimation of the overhead implied by the proposed generation in the BIST context.

References

- [1] M. Abramovici, M.A. Breuer and A.D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, 1990.
- [2] H.J. Wunderlich and Y. Zorian, *Built-In Self Test (BIST): Synthesis of Self-Testable Systems*, Kluwer Academic Publishers, 1997.

- [3] G.L. Smith, "Model for Delay Faults Based upon Paths", *IEEE Int. Test Conf.*, pp. 342-349, 1985.
- [4] W. Wang and S.K. Gupta, "Weighted Random Robust Path Delay Testing of Synthesized Multilevel Circuits", *IEEE VLSI Test Symposium*, pp. 291-297, 1994.
- [5] P. Girard, C. Landrault, V. Moreda and S. Pravossoudovitch, "An Optimized BIST Test Pattern Generator for Delay Testing", *IEEE VLSI Test Symposium*, pp. 94-99, 1997.
- [6] S. Crépiaux-Motte, M. Jacomino, and R. David, "An Algebraic Method for Delay Fault Testing", *IEEE VLSI Test Symposium*, pp. 308-315, 1996.
- [7] A. Virazel, R. David, P. Girard, C. Landrault, and S. Pravossoudovitch, "Delay Fault Testing: Effectiveness of Random SIC and Random MIC Test Sequences", *European Test Workshop*, Cascais (P), May 2000. *To appear in JETTA*.
- [8] F. Brglez, D. Bryant and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits", *IEEE Int. Symp. on Circuits and Systems*, pp. 1929-1934, 1989.
- [9] D. E. Knuth, *The Art of Computer Programming*, vol. 2, *Seminumerical Algorithms*, Addison-Wesley, Reading (MA), 1969.
- [10] TestGen, version Tg4.1 *User Guide*, Synopsys Inc., 1999.
- [11] R. David, *Random Testing of Digital Circuits: Theory and Applications*, Marcel Dekker, Inc., New York, 1998.

Appendix

Proof of Theorem 1 The proof is made of two parts. First we show that the state of the generator is the same at times $l = 0$ and at $l = 2 \times (2^k - 1)$. Second, we show that there is no period shorter than this value.

1. At $l = 2 \times (2^k - 1)$, the state is the same as at $l = 0$.

The state of the generator is made of the state of the LFSR and of the state of the T flip-flops.

Let N_i denote the number of changes of the variable x_i between $l = 0$ and $l = 2^k - 1$. Since the period of the LFSR is $2^k - 1$, x_i changes again N_i times between $l = 2^k - 1$ and $l = 2 \times (2^k - 1)$ because the sequence generated by the LFSR is the same. Hence, between $l = 0$ and $l = 2 \times (2^k - 1)$, the variable x_i has changed $2 \times N_i$ times; since this number is even, x_i has return to its initial Boolean value. This is true for any variable x_i .

2. There is no period shorter than $2 \times (2^k - 1)$.

First case: $n = 2^m$ (hence $b_j = 1$ for every j). Assume $T_i = R_i$ for $i \neq n$, and $T_n = R_0$. Between $l = 0$ and $l = 2 \times (2^k - 1)$, there are (according to Property 10.2 in [11]):

$$2 \times N_i = 2 \times 2^{k-m} = A \text{ changes of } x_i \text{ for } i \neq n, \text{ and} \quad (13)$$

$$2 \times N_n = 2 \times (2^{k-m} - 1) = A - 2 \text{ changes of } x_n. \quad (14)$$

Let us assume that the period is $2 \times (2^k - 1) / d$ and let $N_i(d)$ denote the number of changes of x_i during this period. From (13) and (14),

$$N_i(d) = A / d \text{ for } i \neq n, \text{ and } N_n(d) = (A - 2) / d. \quad (15)$$

From (15),

$$N_1(d) - N_n(d) = 2 / d. \quad (16)$$

In a period, $N_1(d)$ and $N_n(d)$ must be even numbers. However, from (16), if $N_1(d)$ is an even number, $N_n(d)$ can be an even number only if $d = 1$.

Second case: $n \neq 2^m$, and 1) $T_j = R_0 + R_1 + \dots$, and 2) there is T_k such that the number of terms in T_j and T_k is the same. The proof is similar to the first case since $N_j(d) - N_k(d) = 2 / d$. □

We conjecture that the condition $b_j = b_k$ is not necessary, but we have not found a complete proof.