

## 11

## Built-In Self-Test

## 11.1 Introduction

In Chapter 8, we justified the need for DFT techniques. The motivation for the development of the built-in self-test (BIST) technique arose particularly from the cost of test pattern generation and the volume of data that keeps increasing with circuit size. In addition, BIST enables testing at speed. This aspect of testing is especially relevant for present technology, where the effects of the interconnect are causing delays that previously were ignored. Originally, the SAF model was used in conjunction with BIST. Recently, however, this testing is also used to uncover defects that cannot be represented by SAF.

Although there are many BIST schemes, in general, any of those techniques encompasses test pattern generation, test application, and response verification. There are several schemes to generate the test set and to verify the response. These schemes are customized for the type of circuit under test (CUT) and the testing environment. The methodology used for memory testing, *memory BIST*, is different from that used with random logic circuits, *logic BIST*. A combination of these methodologies may be used with RAM-based FPGAs. In this chapter we concentrate on logic BIST. BIST for memories and FPGAs is discussed in Chapters 12 and 13 respectively.

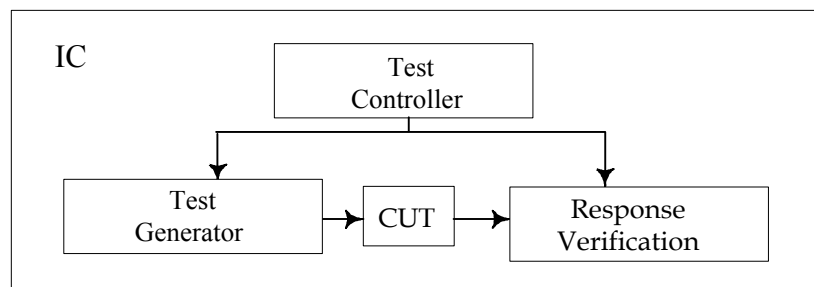


Figure 1. General BIST Architecture

Logic BIST uses mostly pseudorandom (PR) tests. These tests are generated using a Linear Feedback Shift Register (LFSR) or cellular automata. They are usually much longer than deterministic tests, but are definitely less costly to generate. A control signal may trigger the test pattern generation at will. Because of the large volume of test patterns, some type of compactor is used to reduce the response data. There are several compacting schemes, but

signature analyzers are the most popular. Both the random test generator and the compactor are *built* on the chip, as shown in Fig. 11.1. A test controller is added to manage application of the test.

We first describe pseudorandom test generation—the mathematical foundation of LFSRs and the hardware implementations. This is followed by response compaction. Both time and space compaction are explained using LFSRs. One major drawback of compaction is the loss of information. Although the test applied has more than one pattern to detect the fault, the compaction yields results that are identical to the fault-free response. This property is known as *aliasing*. Faults that are hard to detect with PR tests are called *random pattern resistant* (RPR) and are addressed in Section 11.4. Section 11.5 is devoted to BIST architectures for general logic ICs. Some of these are configurations of BIST with other DFT constructs.

## 11.2 Pseudorandom Test Pattern Generation

In random test pattern generation, a pattern may be repeated several times in the process. However, using pseudorandom yields random patterns without repetition. This is equivalent to selection without replacement. The length of the test generated in such a manner depends on the seed of the random number generator. It is conceivable to use some software tools to generate the random patterns and then store them in a ROM that is placed on the chip as illustrated in Fig. 11.1. Two main types of hardware may be used instead: a *cellular automaton* [Hortensius 1989, Bardell 1990] or an LFSR [Bardell 1987].

### 11.2.1 Linear Feedback Shift Register

Examples of LFSRs are shown in Fig. 11.2. For simplicity we use only three- and four-long LFSRs without loss of generality. As its name implies, the LFSR is a shift register with feedback from the last stage and other stages. Besides the clock, it has no other inputs. This is why it is also referred to as *autonomous LFSR* (ALFSR). The outputs of the flip-flops form the test pattern. For this example, there are 3-bit test patterns. The number of unique test patterns is equal to the number of states of the circuit, which is determined, by the number and locations of the individual feedback tabs.

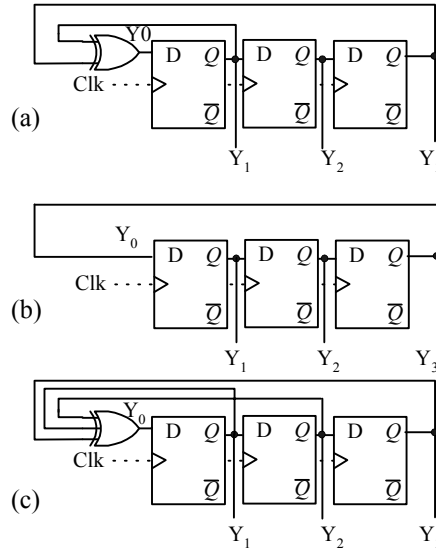


Figure 11.2. An Autonomous Linear Feedback Shift Register For Pseudo-random Test Pattern Generation

Table 11.1. Pseudo-random Pattern Generated by LFSRs in Figure 11.2.

(a)					(b)					(c)				
Clk	Y0	Y1	Y2	Y3	Clk	Y0	Y1	Y2	Y3	Clk	Y0	Y1	Y2	Y3
	1	0	0	1		1	0	0	1		1	0	0	1
1	1	1	0	0	1	0	1	0	0	1	1	1	0	0
2	1	1	1	0	2	0	0	1	0	2	0	1	1	0
3	0	1	1	1	3	1	0	0	1	3	0	0	1	1
4	1	0	1	1						4	1	0	0	1
5	0	1	0	1										
6	0	0	1	0										
7	1	0	0	1										

In Fig. 11.2a, the parity of all feedback leads is the input to the register. Thus we have  $Y_0 = Y_1 \oplus Y_3$ ,  $Y_1 = y_0$ ,  $Y_2 = y_1$ , and  $Y_3 = y_2$ , where  $y_1y_2y_3$  represents the present state and  $Y_1Y_2Y_3$  is the next state of the register. The LFSR was arbitrarily initialized to 001, which is the first pattern appearing on the LFSR. Next we clock the circuits as many times as it is necessary to reproduce this first pattern as illustrated in Table 11.1. In all, there are seven patterns, since the all-zeros pattern cannot be generated. This LFSR produces a *maximal cycle*. Had the feedback been only from the last stage, or from all stages as illustrated in Fig. 11.2b and c, it is easy to determine that the cycle length would have been reduced to only three and four patterns, respectively. The corresponding patterns for the three cases are also shown in Table 11.1.

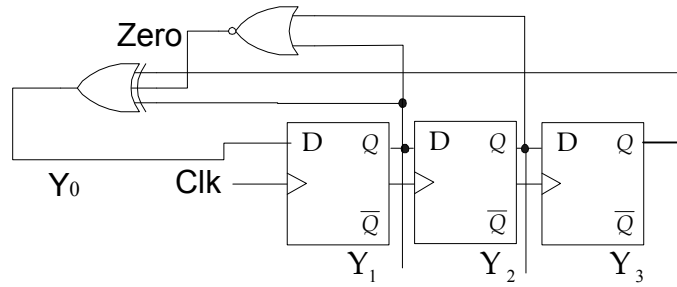


Figure 11.3. The LFSR of Fig. 2 with Adjustment for All 0's Pattern

Table 11.2. Pseudo-random Pattern Generated by LFSR of Figure 11.3

Clk	Y0	Zero	Y1	Y2	Y3
	0	1	0	0	1
1	1	1	0	0	0
2	1	0	1	0	0
3	1	0	1	1	0
4	0	0	1	1	1
5	1	0	0	1	1
6	0	0	1	0	1
7	0	0	0	1	0
8	0	1	0	0	1

The design has been modified by adding a NOR gate to the circuit in Fig. 11.3 to allow the inclusion of the all-zeros pattern. The results are shown in Table 11.2, where eight distinct patterns are listed. The output of the LFSR can also be tapped from the last flip-flop only. The sequence obtained will then be the last column of sequences in Table 11.1 or 11.2. This sequence has some properties that are dependent on the connectivity of the LFSR.

### 11.2.2 LFSR Configurations

Table 11.3. Modulo 2 Operations

$a$	$b$	$a \oplus b$	$a$	$b$	$a + b$ sum	$a + b$ carry	$a$	$b$	$a - b$ difference	$a - b$ borrow
0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	1	1	0	0	1	1	1
1	0	1	1	0	1	0	1	1	0	0
1	1	0	1	1	0	1	1	0	0	0

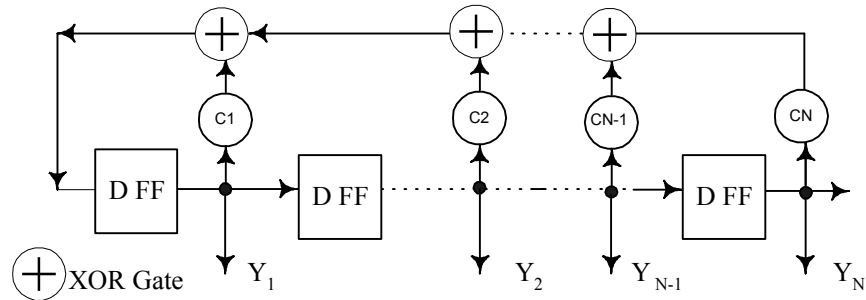


Figure 11.4. A Generic LFSR with possible taps from all Flip-flops

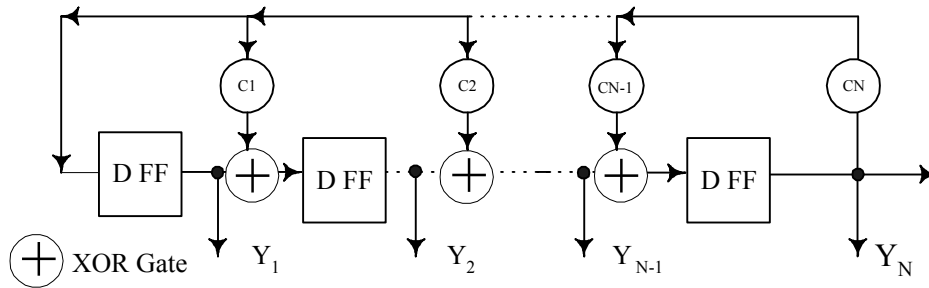


Figure 11.5 A Generic LFSR with possible Taps from all Flip-flops

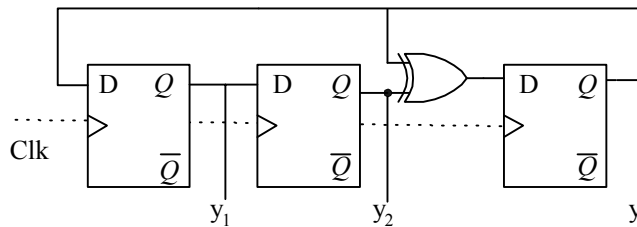


Figure 11.6. The Modular Configuration of the LFSR in Fig. 11.2

The circuit in Fig. 11.4 shows a generic representation of a *standard* LFSR. There is a feedback from the  $i$ th D flip-flop if  $C_i = 1$ ; otherwise, the output of this flip-flop is not tapped. There is another configuration of the LFSR in which the XOR gates are inserted between the flip-flops. This *modular* arrangement is shown in Fig. 11.5. The  $C$ 's values represent the same meaning as for the standard configuration. Also, the XOR gate at the input of flip-flop 1 is not necessary because there is only one input, the output of the  $n$ th flip-flop. The output sequence at the last stage is a function of the initial seed in the LFSR and the feedback coefficients,  $C_i$ . The modular LFSR equivalent to the

LFSR in Fig. 11.2a is shown in Fig. 11.6. Another example for standard and modular LFSRs is shown in Fig. 11.7 for the polynomial  $1 + X^3 + X^4$ .

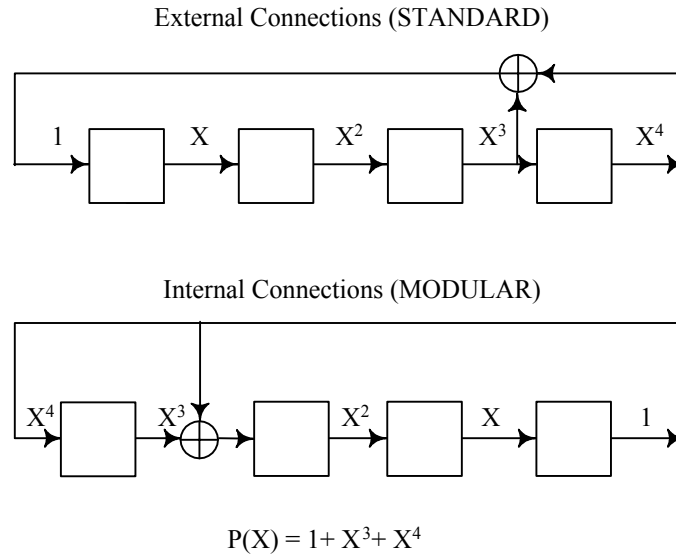


Figure 11.7. Standard and Modular Configurations for the Same Polynomial.

### 11.2.3 Mathematical Foundation of LFSR

First, we discuss the XOR operation. The truth table for this function is shown in Table 11.3. It represents the addition and difference, modulo 2, which are also defined in the same table. In these definitions we include the carry and the borrow, just for clarification, but since the operations are modulo 2, both terms are not part of the results of the operations. In the next derivation we use the plus sign to indicate an XOR operation, addition or subtraction, according to the context.

As a function of time,  $Y_j$  can be represented for the standard form as

$$Y_j(t) = Y_{j-1}(t - 1) \quad \text{for } j \neq 0 \tag{11.1}$$

This is a translation in time of the value at the flip-flop preceding it. We can thus express  $Y_j$  in terms of  $Y_0$  as

$$Y_j(t) = Y_0(t - j) \tag{11.2}$$

If we denote the translation operator as  $X^k$ , where  $k$  represents the time translation units, we can write the Eq. 11.2 in the form

$$Y_j(t) = Y_0(t)X^j \tag{11.3}$$

On the other hand,

$$Y_0(t) = \sum_{j=1}^{j=N} C_j Y_j(t) \quad 11.4$$

where the summation is equivalent to an XOR operation. Then substituting Eq. 11.3 in Eq. 11.4, we get

$$Y_0(t) = \sum_{j=1}^{j=N} C_j Y_0(t)X^j \quad \text{for } 1 \leq j \leq N \quad 11.5$$

Because of the linearity property, we can rewrite Eq. 11.4 as

$$Y_0(t) = Y_0(t) \sum_{j=1}^{j=N} C_j X^j \quad 11.6$$

and

$$Y_0(t) \left\{ \sum_{j=1}^{j=N} C_j X^j + 1 \right\} = 0 \quad 11.7$$

We can then write this expression as  $Y(t) P_N(X) = 0$  11.8

For non-trivial solutions,  $Y_0(t) \neq 0$ , then we must have  $P_N(X) = 0$ . 11.9

where,  $P_N(X) = 1 + \sum_{j=1}^{j=N} C_j X^j$  11.10

$P_N(X)$  is called the *characteristic polynomial* of the LFSR.

We illustrate the use of this polynomial for the 3-bit LFSRs ( $N = 3$ ) shown in Fig. 11.2. For the first one, we have  $C_3 = 1$ ,  $C_2 = 0$ , and  $C_1 = 1$ . Thus the characteristic polynomial is

$$P_3(X) = X^3 + X + 1. \quad 11.11$$

For the second LFSR,  $C_3 = 1$ ,  $C_2 = 0$ , and  $C_1 = 1$ , and the  $P_3(X) = X^3 + 1$ . Finally for the third LFSR,  $C_3 = 1$ ,  $C_2 = 1$ , and  $C_1 = 0$ , which results in  $P_3(X) = X^3 + X^2 + X + 1$ . Using similar analysis on the modular LFSR shown in Fig. 11.6 will result in the same polynomial given by Eq. 11.11.

**11.2.3.1 Reciprocal Polynomial.** The reciprocal polynomial of  $P(X)$  is defined by

$$P_N^R(X) = X^N P_N(1/X) = X^N \left\{ 1 + \sum_{j=1}^{j=N} C_j X^j \right\} \quad 11.12$$

$$P_N^R(X) = X^N + \sum_{j=1}^{j=N} C_j X^{N-j} \quad \text{for } 1 \leq j \leq N \quad 11.13$$

Thus every coefficient  $C_i$  in  $P(X)$  is replaced by  $C_{N-i}$ . For example, the reciprocal of polynomial given by Eq. 11.1 is  $P^R_3(X) = 1 + X^2 + X^3$ . The length of the LFSR sequence is determined by its characteristic polynomial. An  $N$ -degree polynomial yields a maximal length cycle  $2^N - 1$  if it is a *primitive* polynomial. Only a *primitive* polynomial guarantees a maximal-length sequence. But first, let us define operations on polynomials.

**11.2.3.2 Operations on Polynomials.** All operations are modulo 2; thus addition and subtraction are identical according to the definitions in Table 11.3. We illustrate the multiplication and division by examples. For these operations we need to use the *modulo 2* addition property:  $x^i + x^i = 0$ . Consider the fourth degree polynomial,  $S(x) = x^4 + x^3 + 1$ . We will multiply it by the first degree, polynomial  $x + 1$ , and then divide by the second degree polynomial,  $D(x) = x^2 + 1$ .

$$\begin{array}{r} x^4 + x^3 \quad + 1 \\ \underline{\phantom{x^4 + x^3} \phantom{+ 1} x + 1} \\ x^4 + x^3 \quad + 1 \\ \underline{\phantom{x^4 + x^3} \phantom{+ 1} x^5 + x^4 + x} \\ x^5 + \phantom{x^4} + x^3 + x + 1 \end{array} \quad \text{since } x^4 + x^4 = 0$$

Division is of particular interest when LFSRs are used for response compaction.

$$\begin{array}{r} x^2 + x + 1 \\ x^2 + 1 \} \underline{\phantom{x^2 + x + 1} x^4 + x^3 + \phantom{+ 1}} \\ \phantom{x^2 + 1} \underline{\phantom{x^2 + x + 1} x^4 + \phantom{x^3} + x^2} \\ \phantom{x^2 + 1} \phantom{x^4 +} \underline{\phantom{x^2 + x + 1} x^3 + x^2} \\ \phantom{x^2 + 1} \phantom{x^4 +} \phantom{x^3 +} \underline{\phantom{x^2 + x + 1} x^3 + x} \\ \phantom{x^2 + 1} \phantom{x^4 +} \phantom{x^3 +} \phantom{x^2 +} \underline{\phantom{x^2 + x + 1} x^2 + x + 1} \\ \phantom{x^2 + 1} \phantom{x^4 +} \phantom{x^3 +} \phantom{x^2 +} \phantom{x^2 +} \underline{\phantom{x^2 + x + 1} x^2 + x + 1} \\ \phantom{x^2 + 1} \phantom{x^4 +} \phantom{x^3 +} \phantom{x^2 +} \phantom{x^2 +} \phantom{x^2 +} x \end{array}$$

The result of the division of polynomials  $S(x)$  by  $D(x)$  is the quotient  $Q(x) = x^2 + x + 1$  and the remainder  $R(x) = x$ .

**11.2.3.3 Properties of Polynomial**

1. An irreducible polynomial is that polynomial which cannot be factored, and it is divisible only by itself and 1.
2. An irreducible polynomial of degree  $n$  is characterized by:
  - An odd number of terms including the 1 term
  - Divisibility into  $1 + x^k$ , where  $k = 2^n - 1$
3. Any polynomial with all even exponents can be factored and hence is reducible.
4. An irreducible polynomial is *primitive* if the smallest positive integer  $k$  that allows the polynomial to divide evenly into  $1 + x^k$  occurs for  $k = 2^n - 1$ , where  $n$  is the degree of the polynomial.

All polynomials of degree 3 that also include the term 1 are

$$x^3 + 1 = 0$$

$$x^3 + x^2 + 1 = 0$$

$$x^3 + x + 1 = 0$$

$$x^3 + x^2 + x + 1 = 0$$

The second and third polynomials are primitive and it is not difficult to verify that they will divide evenly the polynomial  $x^7 + 1$ . The other two are reducible. As an exercise, you may verify that

$$x^3 + 1 = (x + 1)(x^2 + x + 1)$$

$$x^3 + x^2 + x + 1 = (x + 1)(x^2 + 1)$$

There may be several primitive polynomials of degree  $N$ . However, we are interested in those that include the fewer terms since this means using less XOR gates in the LFSR.

Table 11.4 Primitive Polynomials of Degree

N	Polynomials
4	$1 + X + X^4$
5	$1 + X^2 + X^5$
8	$1 + X^2 + X^3 + X^4 + X^8$ $1 + X + X^5 + X^6 + X^8$
10	$1 + X^3 + X^{10}$
12	$1 + X + X^3 + X^4 + X^{12}$
16	$1 + X^2 + X^3 + X^5 + X^{12}$ $1 + X + X^3 + X^4 + X^{16}$
24	$1 + X + X^3 + X^4 + X^{24}$
32	$1 + X + X^2 + X^{22} + X^{32}$
48	$1 + X + X^{27} + X^{28} + X^{48}$

Table 11.4 lists primitive polynomials of degree  $N$ , where  $1 \leq N \leq 48$  [Peterson 1972, Golomb 1982, Bardell 1987]. However these polynomials are not unique. Notice that for a given  $N$  there may be more than one primitive polynomial with the same number of terms as indicated for  $N = 8$  and  $N = 16$ . Any primitive polynomial includes the terms 1 and  $X^N$ . All these polynomials correspond to LFSRs with at most three taps (XOR gates). This is true also for primitive polynomials of degrees up to 300 [Bardell 1987].

### 11.2.3.4 Properties of Maximal-Length Sequence

1. The maximal-length shift register sequence has a period of  $2^N - 1$ .
2. If the sequence associated with an  $n$ -stage LFSR is of length  $2^n - 1$ , the characteristic polynomial associated with the LFSR is primitive.
3. The numbers of 1's in a maximal-length sequence differ from the number of 0's by 1.
4. Any maximal sequence produces a run of  $n$  1's and  $n - 1$  of 0's.
5. For  $n > 4$ , the maximal sequence for a polynomial  $P_n$  is the reverse sequence of that associated with the reciprocal polynomial  $P_n^R$ .
6. In any such sequences, one-half the runs have length 1, one-fourth have length 2, and so on.
7. The sequence obtained at any stage  $j$  is one clock cycle behind the sequence at stage  $j - 1$ .

You can easily verify these properties for the polynomials discussed so far. We next use the example for the fifth order polynomial:  $X^5 + X^2 + 1$  with the initial seed of {00001}. The output of one of the last flip-flop gives the sequence:  $S(t) = \{(10000\ 10010\ 11001\ 11110\ 00110\ 11101\ 0)\ 1000\ 01\}$ . For the reciprocal LFSR,  $S^R(t)$  includes several runs of different lengths: for example, one run of five and four 1's, two runs of length 3 and length 2, and five runs of length 1.

This distribution shows that the patterns produced have a predetermined distribution of grouping bits [Rajski 1998]. Also, the sequences from the different stages are autocorrelated. For example, the sequences for the second and third stages of the LFSR in Fig. 11.2 and given in Table 11.1 are

(1 0 0 1 1 1 0) 1 ...

(0 0 1 1 1 0 1) 0 ...

Because of these facts, pseudorandom patterns may fail to detect some faults. Such faults are called *random pattern-resistant* (RPR) faults and are discussed in Section 11.4.

## 11.3 Response Compaction

It is necessary to compress the responses in BIST since the interest is to check the response of the test on a chip. In such a manner it is possible to obtain a signature for the circuit at the end of the test application. Comparing the faulty and fault-free signatures will allow us to detect faults. Usually we think of data compression as a process that

preserves data integrity. This is why we give more attention here to data compaction, which may result in some losses. For example, if the signature of a faulty circuit, due to the compaction scheme, is equal to the fault-free signature, the fault escapes detection. This type of information loss is called *aliasing*. The probability of aliasing decreases as the length of the test increases.

There are several compaction testing techniques: (1) parity testing, (2) one counting, (3) transition counting, (4) syndrome calculation and, (5) signature analysis. We describe briefly the first three techniques and illustrate them using single-output circuits. Signature analysis is described in Section 11.3.4. Subsequently, we consider multiple-output circuits in the section on *space compaction*.

### 11.3.1 Parity Testing

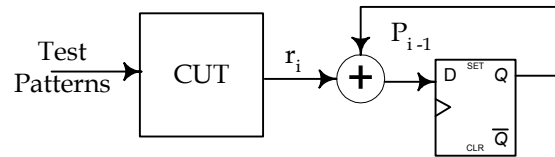


Figure 11.8 Parity Testing

This is the simplest of all techniques but also the most lossy. The parity of responses to the test patterns is calculated

as,  $P = \sum_{i=1}^{i=L} r_i$ , where  $L$  is the length of the test and  $r_i$  is the response for the  $i$ th test pattern. The summation here is

modulo 2. The response of the circuit under test (CUT) to patterns  $i$  and the partial product  $P_{i-1}$  are as illustrated in Fig. 11.8. The result on the storage device after the last test is applied is the parity to be compared to the fault-free circuit parity. If an odd number of test patterns detect the fault, the parity of the faulty circuit is different from that of the fault-free one and the fault is definitely detected. Otherwise, the number is even, and the parity is indistinguishable from the fault-free parity.

### 11.3.2 One Counting

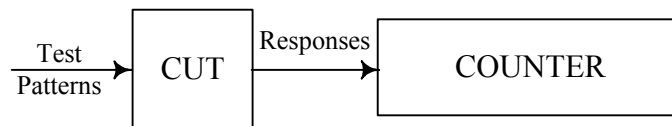


Figure 11.9-Counting Compaction Scheme.

In this scheme, which is shown in Fig. 11.9, the number of 1's in the response stream is calculated and compared to the number of 1's in the fault-free responses [Akers 1989]. We refer to this number as a 1-count. The responses to the test are applied to a counter that counts up, each time the response  $r_i = 1$ .

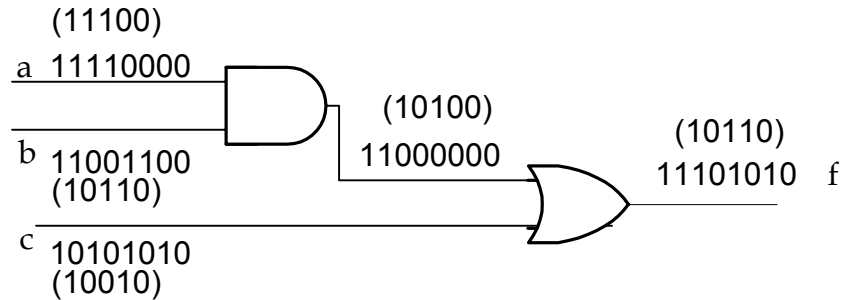


Figure 11.10 Circuit to Illustrate 1-Counting

Consider the circuit in Fig. 11.10, on which we applied an exhaustive test consisting of eight patterns. For the fault-free circuit, 1-count = 5, notice that this number represents the number of minterms of the function. Why? For  $a/0$  and  $a/1$ , the counts are 4 and 6, respectively. Both faults will thus be detected. A fault that causes equal changes of 1's to 0's and vice versa, will have the same number of 1's as in the fault-free count. If we have a test of length  $L$  and the fault-free count is  $m$ , the possibility of aliasing is  $[C(L, m) - 1]$  patterns out of a total number of possible strings of length  $L$ ,  $(2^L - 1)$ . Thus the upper limit of the aliasing probability is  $\text{Pa}(m) = [C(L, m) - 1]/(2^L - 1)$ . However, because the distribution of  $C(L, m)$  is symmetrical and has a peak at  $L/2$ , this probability is lower for small and large values of  $m$ . Also, this probability is smaller the larger  $L$  is.

For the example in Fig. 11.10, where  $m = 5$  and  $L = 8$ , this probability will be  $\text{Pa}(m) = 55/255 \cong 0.2$ . However, it will be left to the problems to show that this test will not cause any aliasing. Notice also that not all of the 255 strings of length 8 will actually be generated since there are only as many strings as there are faults. In this case we have only 10 faults.

Consider a shorter test, say, the pseudoexhaustive test shown in parentheses on the circuit. The good circuit count is 3 and the test length is 5. The aliasing probability is then  $10/31 \cong 0.3$ . Actually, aliasing happens only for one fault,  $a/0$ . For this fault as well as the fault-free circuit, 1-count = 3. Faults which without compaction are detected by an odd number of patterns will always be detected by 1-counting. Only some of those detected by an even number of patterns will have a 1-count that is different from the fault-free response. In addition, the count is unchanged if the output is inverted, and independent of the order of pattern application for a combinational circuit.

The aliasing probability was estimated assuming that all possible strings will be generated by test application. However, this is exaggerated since the number of faults will probably be lower in numbers. The probability that the fault-free circuit will produce  $m$  1's is only  $P_m = C(L, m) / 2^L$ . Thus the aliasing probability is only  $P_a = P_a(m) P_m = [C(L, m) - 1] / (2^L - 1) \cdot C(L, m) / 2^L$ .

### 11.3.3 Transition Counting

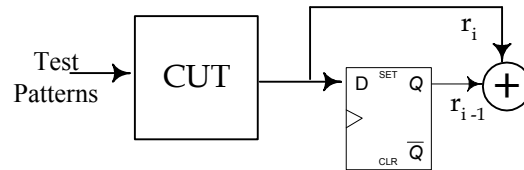


Figure 11.11 Transition Count Compaction Scheme

In transition counting compaction, it is only the number of transition  $0 \rightarrow 1$  and  $1 \rightarrow 0$  that are counted. Thus the

signature is given by  $\sum_{i=1}^{L-1} r_i \oplus r_{i+1}$ , where the summation is ordinary addition and  $\oplus$  is XOR operation. The

compaction scheme is shown in Fig. 11.11.

### 11.3.4 Signature Analysis

Signature analysis is the most popular technique for response compaction and it was originally called *cyclic code checker* [Benowitz 1975]. The technique was then applied for field testing of electronic equipment by Hewlett-Packard electronics [Frohwerk 1977] and became known as *signature analysis*. The compactor in this case is an LFSR to which the response string,  $M(t)$ , from the CUT is applied. At the end of the test application, the content of the LFSR is called the *signature* of the circuit. It reduces the response of all  $L$  test patterns to  $N$ -bit, where  $N$  is the length of the LFSR; that is,  $N$  is the number of flip-flops in the shift register. A fault in the circuit is expected to produce a signature that is different from that of the good circuit. The loss of information due to compaction may cause a faulty circuit to produce a good signature that results in an *aliasing*.

We will first take an example of a test response, a string  $M(t)$ , and generate the signature of this response. We will then show how the response is represented as a polynomial  $M(X)$  and the signature is just the remainder of its division by the characteristic polynomial of the LFSR,  $P(X)$  [Bhavsar 1981]. In addition, we show how aliasing may occur and estimate its probability. Assume that the application of an eight-pattern test to a circuit results in the data stream  $M(t) =$

{10110001}. The leftmost bit is the first signal entering any of the LFSRs shown in Fig. 11.12. The first is of the standard type and its polynomial is  $X^3 + X^2 + 1$  and the second is of the modular type and characterized with the same polynomial. The initial state of the storage elements in the shift register is assumed to be {000}. As this stream of data is shifted in the LFSR, the storage elements change their states following the timing shown in Table 11.5.

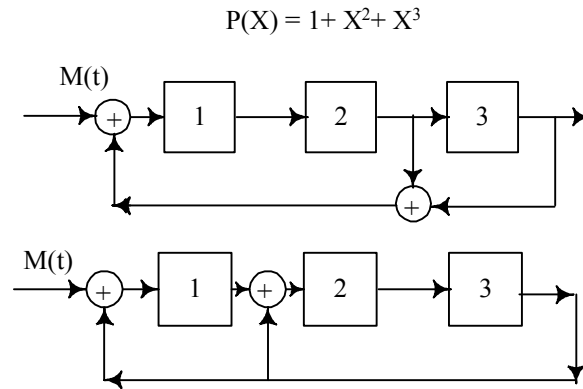


Figure 11.12 Signature Analysis Using Standard and Modular LFSR

Table 11.5 Signature Analysis

T	M(t)	Standard ( $X^3+X^2+1$ )				Modular ( $X^3+X^2+1$ )					Modular ( $X^3+X+1$ )				
		y0	y1	y2	y3	y0	y1	D2	y2	y3	y0	y1	y2	D2	y3
0	1	1	0	0	0	1	0	0	0	0	1	0	0	0	0
1	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0
2	1	0	0	1	0	1	0	0	1	0	1	0	1	1	0
3	1	0	0	0	1	0	1	0	0	1	0	1	0	1	1
4	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1
5	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
7	1	1	0	0	0	1	1	0	0	0	0	0	0	1	1
8	X	X	<b>1</b>	<b>0</b>	<b>0</b>	x	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	x	<b>0</b>	<b>0</b>	<b>1</b>	<b>1</b>
						$R(X) = X^2$					$R(X) = 1$				

We selected one standard LFSR, a modular one with the same characteristic polynomial and a modular one with the reciprocal polynomial as indicated in the figure. The three signatures are {100}, {100}, and {001}, respectively. The signatures on the modular LFSRs are the result of the polynomial divisions of  $M(X)$ , which represents the response string,  $M(t)$ , by the characteristic polynomials of the LFSRs.

The data stream,  $M(t)$ , can be represented by a polynomial in a fashion similar to the representation of the output of an autonomous LFSR in Section 2.3. Each bit is translated in time by one unit. Again, Eq.11.5 defines the translation operator. Thus the data stream,  $M(t) = \{10110001\}$ , is represented by the polynomial  $M(X) = X^7 + X^5 + X^4 + 1$ . The characteristic polynomial of the first modular LFSR in Fig. 11.12 is the same as the standard one,  $P(X) = X^3 + X + 1$ . Notice, however, that the taps are the reciprocal of those of the standard configuration. The division of  $M(X)$  by  $P(X)$  is performed in the same manner as presented in Section 11.2.3 and the results of the operation are a quotient  $Q(X)$ , and a remainder  $R(X)$  such that  $M(X) = P(X)Q(X) + R(X)$ .

$$X^7 + X^5 + X^4 + 1 = (X^4 + X^3 + 1)(X^3 + X^2 + 1) + X^2$$

$Q(X)$	$X^3 + X^2 + 1$		$X^7 + X^5 + X^4 + 1$	+1	$1101$		$11001$	$10110001$
			$X^7 + X^6 + X^4$				<u>1101</u>	
			$X^6 + X^5$	+1			$110001$	
			$X^6 + X^5 + X^3$				<u>1101</u>	
			$X^3 + 1$	+1			$1001$	
			$X^3 + X^2 + 1$				<u>1101</u>	
$R(X)$			$X^2$				$0100$	

The division can also be performed using the coefficients  $C_k$  that we introduced in Section 11.2.3.2. As we recall,  $C_k = 1$  only when there is feedback from the last stage to the  $k$ th flip-flop; otherwise it is 0 for the other modular LFSR. The taps are the same as for the standard LFSR, but the polynomial is the reciprocal of the latter's polynomial.

As another example, we divide  $M(X)$  by  $P(X) = X^3 + X + 1$ .

$$X^7 + X^5 + X^4 + 1 = X^4(X^3 + X + 1) + 1$$

$Q(X)$	$X^3 + X + 1$		$X^7 + X^5 + X^4 + 1$	$1011$		$1$	$10110001$
			$X^7 + X^5 + X^4$				<u>1011</u>
$R(X)$			$1$				$0001$

This remainder corresponds to the signature  $\{001\}$  left in the corresponding LFSR as indicated in Table 11.5.

**11.3.4.1 Fault Detection.** Consider a CUT and a test set of length  $L$  such that there is at least one pattern that detects any fault,  $f$ . We will apply the response of the faulty circuit to an LFSR that we will call a *signature analyzer*, as we use it in compaction. This signature analyzer has  $N$  stages and a characteristic polynomial,  $P(X)$ . The fault  $f$  is detected through the signature analyzer if the faulty signature  $S_f$  is such that  $S_f \oplus S = 1$ , where  $S$  is the signature of the fault-free circuit; otherwise, the fault escaped detection. It is masked and we say that *aliasing*

has occurred. In terms of polynomial division, aliasing implies that the division of the faulty and faulty-free responses,  $M_f(X)$  and  $M(X)$ , yield the identical remainders. That is,  $R_f(X) = R(X)$ .

Instead of analyzing the response stream, we consider the error,  $E(X)$ .

$$M(X) = P(X) Q(X) + R(X)$$

$$M_f(X) = P(X) Q_f(X) + R_f(X)$$

Adding the two equations we get

$$E(X) = P(X)q(X) + r(X)$$

The fault is detected if  $r(X) \neq 0$ .

For a given polynomial  $P(X)$ , there are only a finite number of remainders, and therefore it is not possible to eliminate aliasing but only to minimize it. For example, if the length of the test set is equal or smaller than the length of the LFSR's cycle,  $L \leq N$ , all faults will be detected. However, most pseudorandom test sets are very long and it is not practical to have a very long LFSR.

**11.3.4.2 Aliasing Probability.** We calculate the aliasing probability for a test of length  $L$  and a signature analyzer of  $N$  stages. Thus any response of the circuit to the test consists of  $L$  bits. It is any of  $k = 2^L$  strings of  $L$ . No aliasing is possible for those response strings with  $k - N$  leading zeros since they are represented by polynomials of degree  $N - 1$  that are not divisible by characteristic polynomial of the LFSR. There are  $2^{L-N}$  such strings. Hence the probability of aliasing is  $\text{Pa} = \{2^{L-N} - 1\} / \{2^L - 1\}$  since  $L \gg 1$  and  $L \gg N$ , then  $\text{Pa} = 2^{-N}$ . This value of the aliasing probability is reached for test lengths is 50 patterns. Thus the larger the  $N$ , the smaller the aliasing probability. Notice that this probability is based on the assumption that the response streams are equally likely to occur. That is, the number of faults is actually  $2^L - 1$ . However, there are generally fewer faults.

Several approaches have been proposed to minimize aliasing—reversing the test sequence, using multiple MISR, taking multiple signatures. It is possible to double the test length by applying the reverse sequence. This can be achieved by changing the LFSR polynomial to its reciprocal. The aliasing probability will be reduced to  $2^{-2N}$ , however, this requires some hardware overhead to reconfigure the LFSR and doubles the test application time. Another approach is to use several LFSRs with polynomials that are relatively prime. An orthogonal practice is to use the same LFSR and take signatures at several time intervals. For  $w$  signatures, the probability will then decrease to  $2^{-wN}$ . More storage is then needed for the various signatures. An extreme case of this scheme is to take a signature

every  $N$  cycles. This scheme would have an aliasing probability of zero. The proof is straightforward using polynomial division and is left as an exercise.

### 11.3.5 Space Compaction

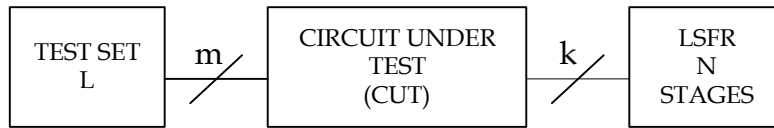


Figure 11.13 Compaction Using Signature Analysis.

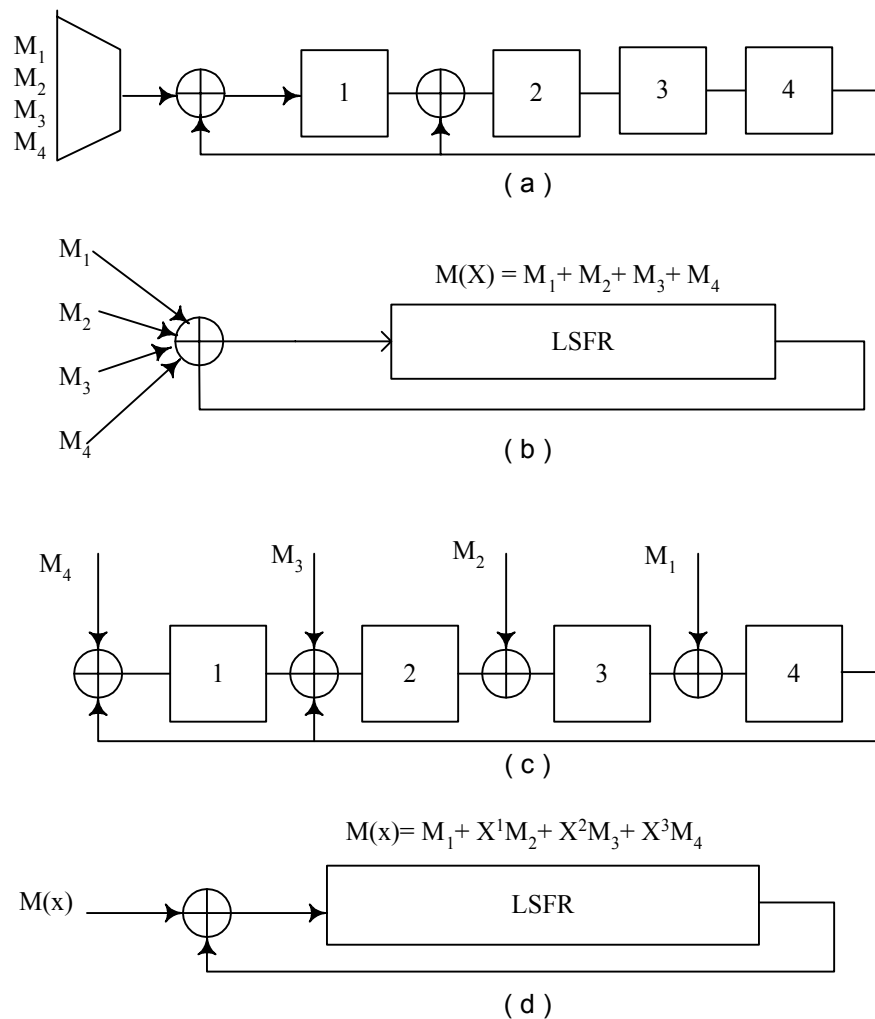


Figure 11.14 Space Compaction Schemes

So far we assumed that the CUT has one primary output. This is not realistic and we need to deal with compaction schemes for  $k$ -output circuits as illustrated in Fig. 11.13. Different possible space compaction schemes are shown in

Fig. 11.14 for a case where  $N = 4$ ,  $L = 5$ , and  $k = 4$ . The characteristic polynomial of the signature analyzer is  $P(X) = X^4 + X^3 + 1$ .

**11.3.5.1 Single-Input Signature Analyzer (SSA).** Responses from one primary output at a time are compacted through the signature analyzer using an n-to-1 multiplexer as illustrated in Fig. 11.14a. This process is slow, but it reduces the aliasing probability since the scheme is equivalent to elongating the test by a factor of  $N$ . The probability of aliasing is still  $2^{-N}$ , however, faults may be detected through more than one output and thus having reduced aliasing.

**11.3.5.2 Bellmac.** This approach, which is featured in Fig. 11.14b, takes its name from the product in which it was used for the first time. In this case parity and signature analyzer compaction techniques are used. All the responses from the primary outputs are compacted through a parity tree before a second compaction by the analyzer.

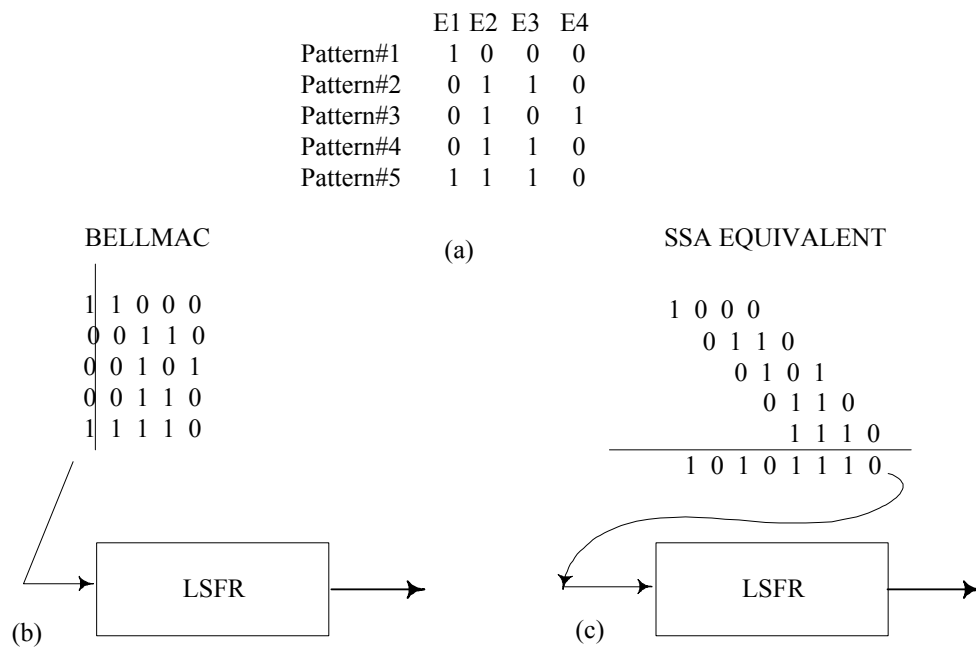


Figure 11.15. Signature Analysis using BELLMAC and PSA

In the example used in this section, the errors at the four outputs of a circuit, in response to five patterns, are shown in Fig. 11.15a. The left most column of Fig. 11.15b shows the parity of these errors for each pattern. The parity stream is then compacted in an LFSR. The polynomial representing this stream of data is  $X^5 + 1$ . Its division by the characteristic polynomial  $P(X) = (X^4 + X + 1)$  results in the remainder  $R(X) = X$ .

**11.3.5.3 Parallel Signature Analysis (PSA).** This scheme is also known as multiple input signature register (MISR) and illustrated in Fig. 11.15c. According to this compaction, the responses from the different primary outputs are introduced at the different stages of the LFSR. Thus the different responses,  $M_i(X)$ , are staggered into the signature analyzer. This scheme is equivalent to compaction through a SSA with the input stream being shifted in time,  $M(X) = M_0(X) + XM_1(X) + \dots + X^n M_n(X)$ . This is illustrated in the example of Fig. 11.15c. The resulting error polynomial due to the four outputs is

$$E(X) = E_1(X) + XE_2(X) + X^2E_3(X) + X^3E_4(X)$$

It is divided by the characteristic polynomial,  $X^4 + X + 1$ , to result in a remainder of  $R(X) = X^3 + X + 1$ . As a consequence of this equivalence, the aliasing probability for MISR is still  $2^{-N}$ , where  $N$  is the number of stages of the signature analyzer.

In the example considered to illustrate MISR, the number of outputs is equal to the number of stages in the signature analyzer. However, we need to determine space compaction when  $n \neq N$ . If  $n < N$ , the outputs will be introduced in  $n$  of  $N$  flip-flops and the other flip-flops will not have any input from the circuit. In the other situation,  $n > N$ , it is possible to use additional LFSRs or to partition the outputs into  $N$  partitions. The parity of each partition is then fed to the different stages of the MISR.

## 11.4 Random Pattern Resistant Faults

The effectiveness of a test set is measured by its fault coverage, its length, and its hardware and data storage requirement. PR tests generated according to the method we described above are usually long and result in unacceptable fault coverage. Figure 11.16a shows a typical plot of the fault coverage versus the number of random test patterns. At first, a rapid increase in fault coverage is followed by saturation. The difference between this saturation level and 100% is denoted by  $\Delta FC$ , and it represents faults that are hard to detect by random patterns.

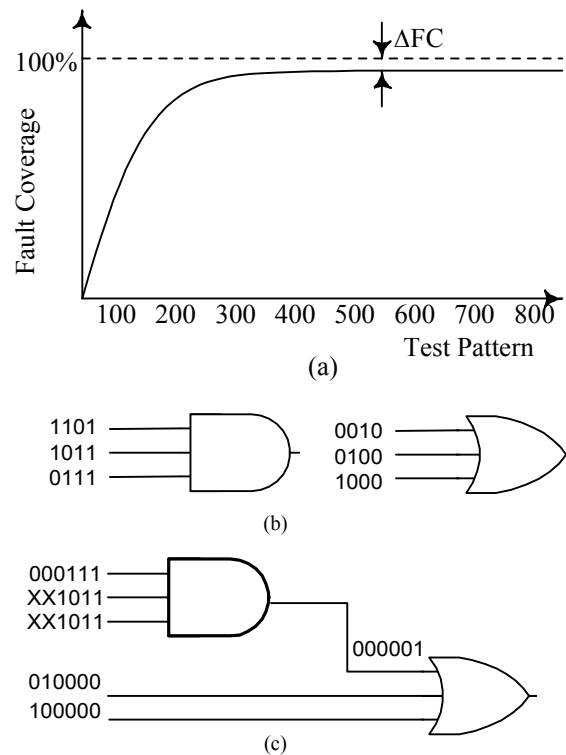


Figure 11.16 RPR Faults: (a) Typical RP Fault Coverage, (b) Weighted PR Test Pattern

They are called random pattern resistant (RPR) faults. The fault coverage can be improved by reducing the aliasing probability. In Section 11.3.4.2 we briefly listed several approaches. However, the low coverage is mostly due to lack of patterns to detect some faults. There are faults that are detected by very few, if not by only one pattern. If this pattern is not part of the test set, the fault will remain undetected. We explain next why some of these patterns are hard to obtain.

PR patterns generation is based on the fact that all flip-flops have equal probability of producing a 1 or a 0. That is, all the inputs of the circuit under test have the same probability for a 0 or a 1. For example, if a six-input NOR gate is considered, SA0 faults on its inputs are detected by one and only one pattern. The probability of generating this pattern is only 1 out of 64 ( $2^6$ ). In addition, the sequence generated is based on an equally likely probability for 0 and 1 to appear at each of the CUT's inputs. Consider the circuit in Fig. 11.16b where the minimal SAF test set is shown for a three-input AND and a three-input OR. More 1's are needed on the AND gate and more 0's are required for the OR gate. For the circuit in Fig. 11.16c, six patterns out of a 32-long exhaustive test set

guarantee 100% fault coverage. It is obvious from the test set that the 0's and 1's do not appear equally on all inputs. Also, inputs  $d$  and  $e$  need to be more exercised than the other primary inputs (PIs). Such an observation prompted the introduction of weights for the various PIs [Schnurmann 1975, Bardell 1987]. The weights are the probabilities for the input signals to be equal to 1. This approach which is called *weighted PR test pattern generation*, improved the fault coverage appreciably, but also added extra hardware.

Several techniques have been used to generate optimal weights. Some are based on circuit structure analysis [Waicukauski 1989], and others are based on fault detection probabilities [Lisanke 1987, Wunderlich 1987]. Despite the improvement resulting from this method, there were still some faults that are hard to detect. For example, going back to Fig. 11.16*b*, if the inputs of the AND and OR circuits are tied together, the weights selected will favor one structure or the other and multiple weights are required. However, this multiple-weight approach requires considerable increase in the hardware.

Another approach to overcome the RPR fault problem is test point insertion. The location of the test points is determined by testability measures. As we discussed in Chapter 8, test point insertion requires knowledge of the circuit topology and extra hardware with possible impact on performance [Schotten 1995]. Reseeding the LFSR is an alternative approach that has low-overhead hardware [Hellebrand 1992]. A similar technique used a multiple polynomial LFSR. However, these techniques generate test sets of excessive length.

Recently, a mixed mode approach has been used. With fault simulation it is possible to determine the RPR faults and use a deterministic test approach to generate the patterns to detect them. BIST execution then combines the PR pattern test ( $R$ ) and the deterministic test ( $T$ ) according to a certain protocol. The test  $T$  has to be stored in a ROM on the chip or implemented as a hardware circuit that is activated at testing time. Bit fixing is one of these techniques. The output bits of the LFSR, at a certain position in the sequence, are changed to generate the RPR test patterns [Pomeranz 1993, Chatterjee 1995, Toubia 1995-1996, AlShaibi 1996]. An alternative to bit fixing is bit flipping [Wunderlich 1996]. Most of these variations of controlling the bits of the LFSR sequence have not yet solved the RPR problem and they were applied to circuits that combine BIST with scan-path. These circuits are described in the next section as part of BIST architectures.

## 11.5 BIST Architectures

Several testing schemes make use of pseudorandom test pattern generation (an LFSR) and response compaction (a signature analyzer). Some of these schemes include both an LFSR and a signature analyzer on chip for internal testing. Others use them off-chip for external testing. Typically logic BIST approaches combine pseudorandom testing with other DFT constructs such as scan path and Boundary-Scan. The intent here is to optimize on the advantages of the various DFT constructs.

### 11.5.1 Built-In Self-Testing

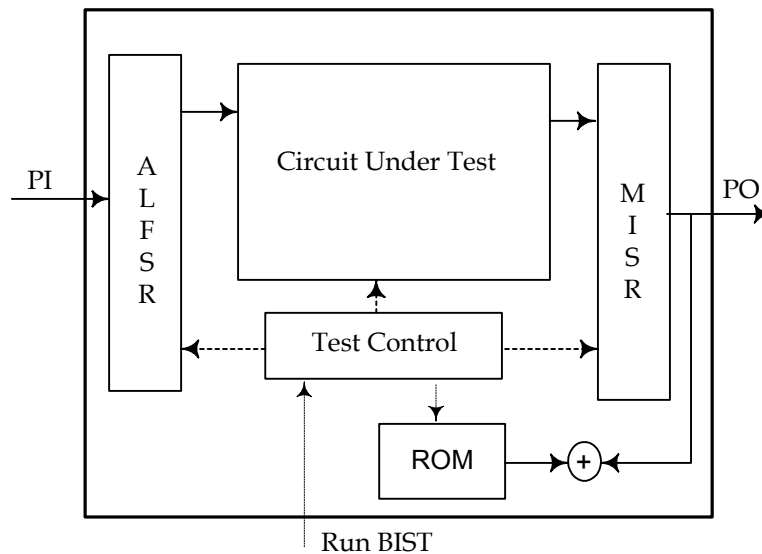


Figure 11.17. BIST Architecture

The built-in self-testing architecture is shown in Fig. 11.17. In addition to the circuit under test (CUT), the chip includes an LFSR for test pattern generation and a MISR for response compaction. Some control circuitry is required to allow for configuring the circuit for normal and testing modes. The signature from the MISR is automatically checked versus the good circuit response, which is stored in a ROM on the chip. Under normal operation the signal is applied to the CUT and observed directly on the primary output. During test mode, the patterns are applied from the LFSR, the response of the circuit to these patterns is compacted through the MISR, and the signature is compared to the good one. The effectiveness of this testing for a given circuit depends on the

appropriate choice of the LFSRs, their length, and configurations. It usually yields high fault coverage for combinational circuits.

### 11.5.2 Autonomous Test

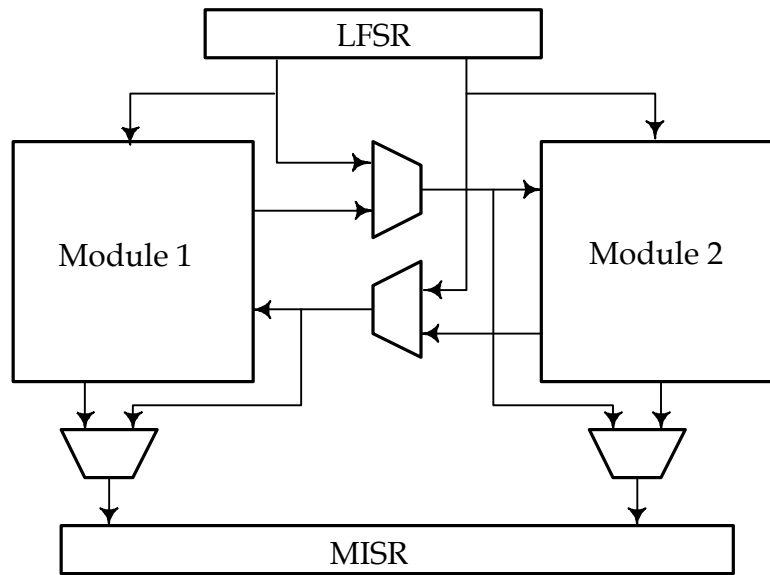


Figure 11.18. Autonomous BIST Testing

In the autonomous test approach [McCluskey 1981], the circuit is partitioned into subcircuits to facilitate its testing, as explained in Chapter 8. The partitioning is accomplished through an added multiplexing scheme or by path sensitization. Each partition is tested independent of the others. The test may be supplied externally, but here we consider built-in testing. Thus the scheme, which is illustrated in Fig. 11.18, uses the same LFSR and the same MISR for test generation and response compaction. The solid lines illustrate the testing of module-1. If the multiplexers are too many for optimal timing of the circuit, a sensitization approach should be taken, although it may take some more effort to configure.

### 11.5.3 Circular BIST

Circular BIST is intended for register-based architecture [Krasniweski 1989]. In Fig. 11.19 the self-test shift registers (STSRs) are part of the circular self-test path. The other registers are not used for testing and they are denoted by SR. Each of the storage devices, STSRs, is designed as shown in Fig. 11.20. The MISR formed by all STSRs has a characteristic polynomial of  $1 + X^m$ , where  $m$  is the number of storage elements in the chain. All these cells should be initializable to put the CUT in a known state before testing. The test process requires three phases:

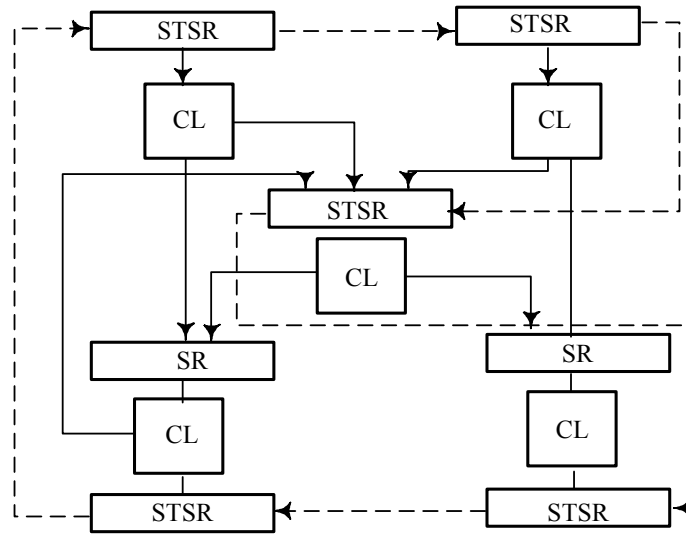


Figure 11.19 Circular BIST

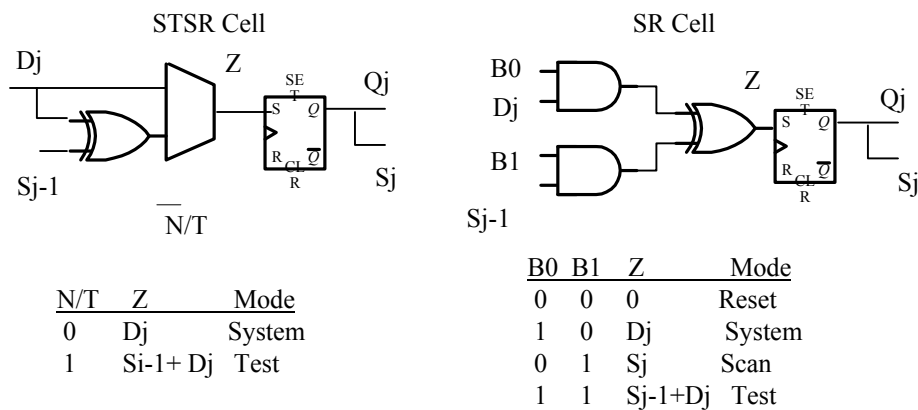


Figure 11.20 Circular BIST: Storage Cell Configurations.

1. Initialization. All registers are placed in a known state.
2. Testing of the CUT. The circuit is run in test mode; registers that are not in the self-test path operate in their normal mode. During this phase the self-test path operates as both an LFSR and a MISR.
3. Response evaluation. During this phase the circuit is also in test mode, but now the sequences of outputs from one or more STSR are compared with a precomputed fault-free value.

### 11.5.4 BILBO

In this and the next subsections, we address the various schemes that combine BIST with scan path. The objective is to optimize on the strength of both techniques to facilitate IC testing. One of the early DFT constructs that combined

BIST and scan path is the built-in logic blocks observer (BILBO) [Konemann 1979-1980].

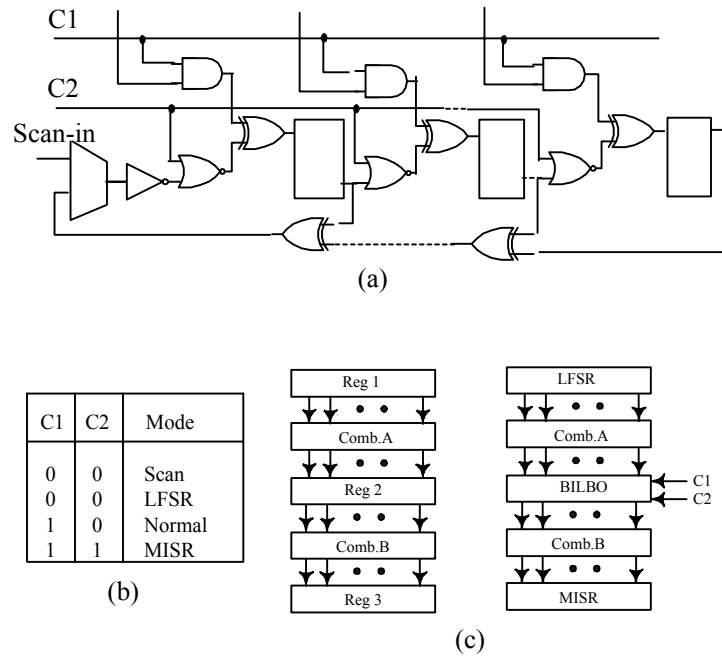


Figure 11.21 BILBO (a) Re-configurable Registers, (b) Modes of Operations, (c) An Example of use of BILBO

This architecture, which combines the functions of test pattern generation and response compression into a single unit, was developed for bus-oriented systems. It takes advantage of registers or storage devices on the chip to configure them as pseudorandom test pattern generators PRTPG and signature analyzers. In Fig. 11.21a the flip-flops can be configured to form any of the following structures: a shift register for scan design, an LFSR, or a MISR according to the mode of operation determined by the two control lines  $C_1C_2$ . A circuit using BILBO is modeled as shown in Fig. 11.21b. The two combinational parts, Comb-1 and Comb-2, are tested, one at a time, using registers R1, R2 and R3. To test Comb-A, R1 is configured as a PRTPG and R2, which is a BILBO block, is used as a MISR. For the Comb-B, it is the BILBO block that is used as a PRTPG and R3 is used as the MISR. These different testing sessions need to be coordinated by a test manager.

### 11.5.5 Random Test Socket

This is a generic architecture that combines scan path and BIST [Bardell 1982] as illustrated in Fig. 11.22. A subsequent modification resulted in more efficient DFT architectures, such as STUMPS, discussed in section 11.5.6. In random test socket architecture, all primary inputs (PIs) to the circuit are connected to the taps of a pseudorandom pattern generator (LFSR1) and all primary outputs (POs) are connected to a MISR. The storage cells of the CUT are

configured for scan-path operations and form the shift register, SR. The SI pin is connected to another pseudorandom pattern generator (LFSR2) and the SO is connected to an SSA. The testing is applied as follows:

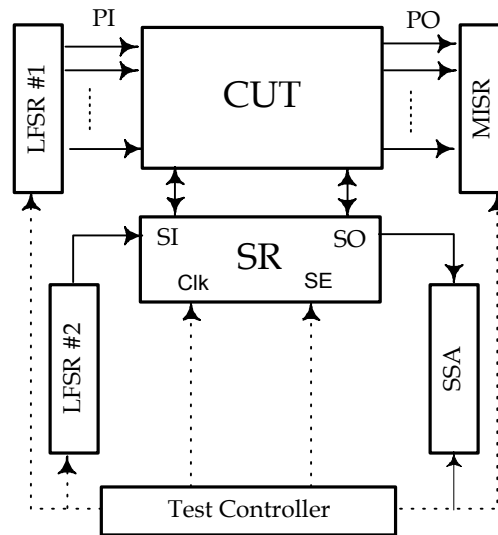


Figure 11.22 Random Test Socket (Bardell 1982)

1. Load the SR with an initialization pattern by clocking the LFSR and SR simultaneously while the scan enable,  $SE = 1$ . On each cycle a new random bit is generated. Thus we need  $n$  clock cycles, where  $n$  is the number of storage units in SR.
2. Apply a test pattern from LFSR 1 to the CUT's PI.
3. Clock the system once with  $SE = 0$  to latch the response to the pattern in SR.
4. Capture the results in MISR by clocking once while keeping  $SE = 0$ .
5. Unload the data of the SR into the SSA by making  $SE = 1$  and clocking both the SSA and SR  $n$  times.

Steps 1 and 5 can be overlapped as we have done in traditional scan path (see Chapter 9). This approach to applying the test pattern in the scan-path registers is called *test per scan* as opposed to *test per clock* used so far in conjunction with BIST.

The advantages of this scheme are low-cost ATPG and improved observability. Also, instead of using two LFSRs for PRTPG, one will generally be used to initialize the scan chain and to stimulate the primary inputs. Similarly, one MISR can be used to compress the response data from the primary outputs and from the scan chain. The limitations are increased overhead and long test application time. In addition, the RPR faults are still a problem for fault coverage.

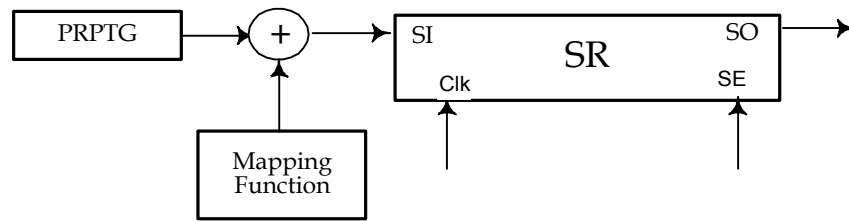


Figure 11.23 Mixed Mode Control for RPR Faults

We illustrate next how to use mixed-mode technique to detect RPR faults. The concept is illustrated in Fig. 11.23. The output of the LFSR is applied to the scan registers as described above, however, these output bits can be changed in such a way as to allow a pattern to detect the RPR faults. Fixing the bits is obtained from the mapping function which was generated using knowledge about the circuit - the test patterns to detect RPR as obtained from a deterministic test pattern generator and are verified by a simulator.

### STUMPS

Self-test using MISR and a parallel shift register sequence generator (STUMPS) architecture was originally used in multichips board in which each board has its SR, which allows the testing of all boards using the same DFT overhead. We discuss it here in term of one chip having multiple scan chains. The scheme is illustrated in Fig. 11.24. The multiplicity of the chains speeds up testing application. The SI pins of all chains are supplied from a multiple-output LFSR. Also all SO pins are connected to the various stages of a MISR.

All SRs are loaded using the shift register sequence generator (SRSG). Both registers are clocked simultaneously. The number of cycles is equal to the longest scan chain of the SRs. The results are first captured back in the SRs, then scanned out while another test is loaded. In the example shown, we have  $m = 4$  and nothing is shown about how the CUT receives test patterns. We can consider two alternatives: use a parallel LFSR or use Boundary-Scan cells. The second solution, of course, make the test controller more complex but it is feasible. This testing architecture requires only one additional I/O pin, the test mode signal. The use of multiple scan chains accelerates test application. The main limitation of the architecture is the need of addition interconnections, which makes routing more difficult.

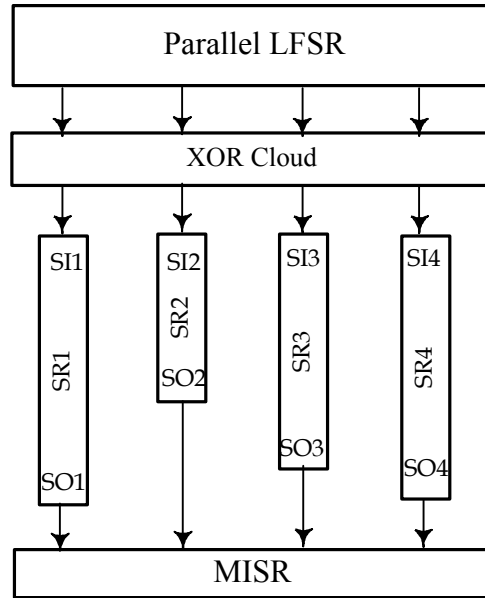


Figure 11.24 STUMPS Architecture (Bardell 1985)

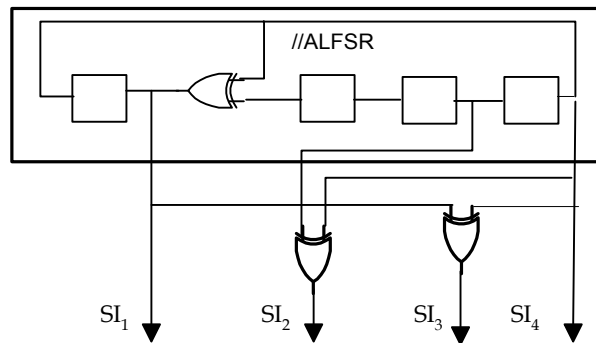


Figure 11.25 Details of XOR Cloud in Fig. 11.24

The sequences obtained from adjacent bits of a parallel LFSR are not linearly independent; the neighboring scan chains contain test patterns that are highly correlated [Chen 1986]. This can affect fault coverage adversely since the patterns seen by the CUT do not really form a random sequence. To alleviate the consequence of this problem, phase shifters are placed between the LFSR and the SI pins of the scan chains [Bardell 1987]. Phase shifters are formed with an XOR network and form the XOR cloud box shown in the figure. An example of the shifters attached to the PRTPG is shown in Fig. 11.25.

## References

- Akers, S. B. and B. Krishnamurthy (1989), Test Counting: a tool for VLSI testing, *IEEE Des. Test Comput.*, Vol. 6 No.10, pp. 58 – 72.
- AlShaibi, M. F. and C. R. Kime (1996), MFBIST: A BIST Method for Random Pattern Resistant Circuits,” *Proc. IEEE International Test Conference*, pp. 176 – 185.
- Bardell, P. H., (1982), “Self-testing of multi-chip logic modules,” *Proc. IEEE Int’l Test Conf.*, pp. 200 – 204.
- Bardell, P.H. et al., (1987), *Built in Test for VLSI: Pseudo-random Techniques*, Wiley, New York.
- Bardell, P. H. (1990), Analysis of Cellular Automata Used as Pseudo-random pattern Generators, *Proc. IEEE International Test Conference*, pp. 762 – 767.
- Benowitz, N. et al. (1975), An advanced fault isolation system for digital logic, *IEEE Trans. Comput.*, Vol. C-24, No. 5, pp. 489 – 497.
- Bhavsar, D. K. and R. W. Heckelman (1981), Self-Testing by polynomial division, *Digest of Papers, International Test Conference*, pp. 181 – 189.
- Chakrabarty, K. et al. (1997), Test with Compression for built-in self testing, *Proc. IEEE International Test Conference*, pp. 328 – 337.
- Chatterjee, M. and D. Pradham (1995), A novel pattern generator for near-perfect fault coverage,” *IEEE VLSI Test Symposium*, pp. 417 – 425.
- Chen, C. L. (1986), Linear dependencies in linear feedback shift registers, *IEEE Trans. Comput.*, Vol. C-35, No. 12, pp. 1086 – 1088.
- Frohwerk, R. A., (1997), *Signature Analysis: A New Digital Field Service Method*, Hewlett Packard Journal, Vol. 28, No. 9, pp. 2 – 8.
- Golomb, S. W. (1982), *Shift Register Sequences*, Aegean Park Press, Laguna Hills, CA.
- Hellebrand, S. et al. (1992), Generation of vector patterns through reseeding of multiple-polynomial linear feedback shift registers, *Proc. IEEE International Test Conference*, pp. 120 – 129.
- Hortensius, P. D. et al. (1989), Cellular automata based pseudo-random number generators for built-in self -test, *IEEE Trans. Comput. Aided Des.*, Vol. CAD-8, No. 8, pp.842 – 859.
- Konemann, B. (1979), Built-in logic block observable technique, *Proc. IEEE International Test Conference*, pp. 37 – 41.
- Koenemann, B. (1980), Built-in test for complex digital integrated circuits, *IEEE J. Solid State Circuits*, Vol. SC-15, No. 3, pp. 315 – 318.
- Krasniweski, A. and S. Pilarski (1989), Circular self-test path: A low cost BIST technique for VLSI circuits, *IEEE Trans. Comput. Aided Des.*, Vol. 8, No. 1, pp. 46 – 55.
- Lisanke, R., F. Brglez, and A. Degeus (1987), Testability-Driven Random Test Pattern Generation, *IEEE Trans. Comput. Aided Des.* Vol. CAD-6, No. 6, pp. 1082 –1087.
- McCluskey E. J. and S. Bozorgui-Nesbat. (1981), Design for autonomous test, *IEEE Trans. Comput.*, Vol. C-30, Vol. 11, pp 860 – 875.

McCluskey E. J. (1986), *Logic Design Principles with Emphasis on Testable Semicustom Circuits*, Prentice Hall, Upper Saddle River, NJ.

Peterson, W. W., and E. J. Weldon (1972), *Error-Correcting Codes*, Colonial Press, Birmingham, AL.

Pomeranz, I. and S. M. Reddy (1993), "3-Weight pseudorandom test generation based on a deterministic test set for combinational and sequential circuits," *IEEE Trans. Comput. Aided Des.*, Vol. CAD-12, No. 7, pp. 1050–1052.

Rajski, J. and J. Tyszer (1998), *Arithmetic built-in self-test for embedded systems*, Prentice Hall, Upper Saddle River, NJ.

Schotten, C. and H. Meyr (1995), Test-point insertion for an area-efficient BIST, *Proc. IEEE International Test Conference*, pp.515 – 523.

Schnurmann et al. (1975), The weighted random test-pattern generator, *IEEE Trans. on Comput.*, Vol C-24, No. 7, pp. 695 – 700.

Touba, N. A. and E. J. McCluskey (1995), Transformed pseudorandom patterns for BIST, *IEEE VLSI Test Symposium*, pp. 2–8.

Touba, N. A. and E. J. McCluskey (1996), Altering a pseudorandom bit sequence for scan-based BIST, *Proc. IEEE International Test Conference*, pp.167–175.

Waicukauski, J. et al. (1989), A method for generating weighted random test patterns, *IBM J. Res. Dev.*, Vol. 33, No. 2, pp. 149–161.

Wunderlich, H. J. (1987), On computing optimized input probabilities for random tests, *Proc. ACM Des. Automation Conference*, pp.392–398.

Wunderlich, H. J. and G. Kiefer (1996), Scan-Based BIST with complete fault coverage and low hardware overhead, *Proc. IEEE European Test Workshop*, pp. 60 – 64.

## Problems

- 11.1.** Show that the polynomial  $P(X) = 1 + X + X^4$  has a maximal test sequence.
- 11.2.** Is the polynomial  $P(X) = 1 + X^5$  primitive? Explain.
- 11.3.** Give, if any, the characteristic polynomial of the LFSR that causes aliasing when the following error sequence is

$$\begin{array}{l} t = 0 \ 1 \ 2 \ 3 \ 4 \ \dots \\ E = 1 \ 0 \ 0 \ 0 \ 0 \ \dots \ 0 \ 0 \ 0 \ 0 \ 0 \end{array}$$

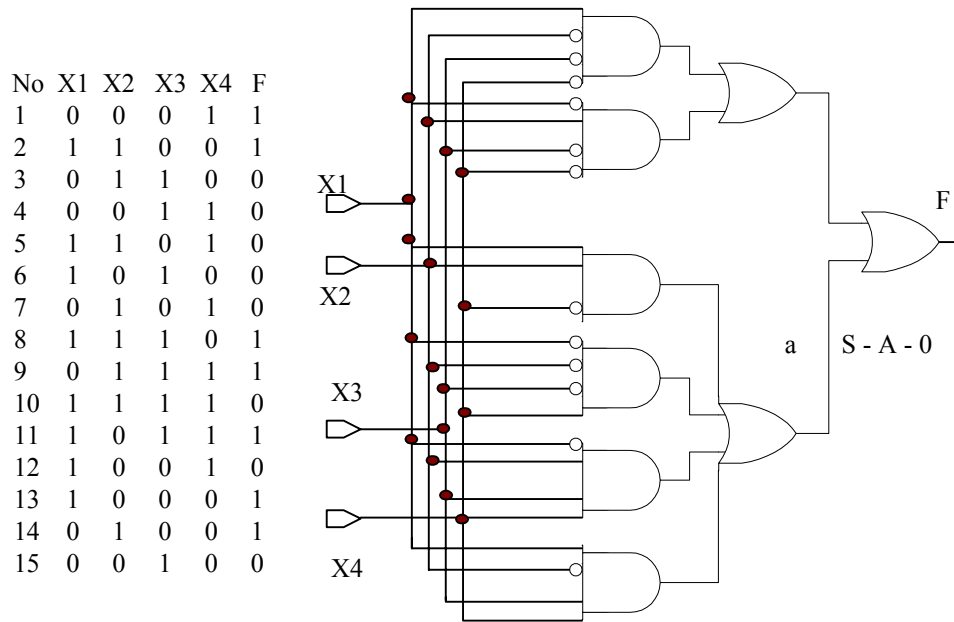


Figure 11.P1

- 11.4.** Suppose that a modular LFSR whose characteristic polynomial is  $1 + X + X^4$  is used as a signature analyzer for the circuit and the test shown in Fig. P11.4.
- (a) Calculate the signature of the good circuit
  - (b) Calculate the signature if the test sequence is 1, 2, 3, ... 15
  - (c) Repeat part (b) with the sequence reversed, 15, 14, 13, ... 1

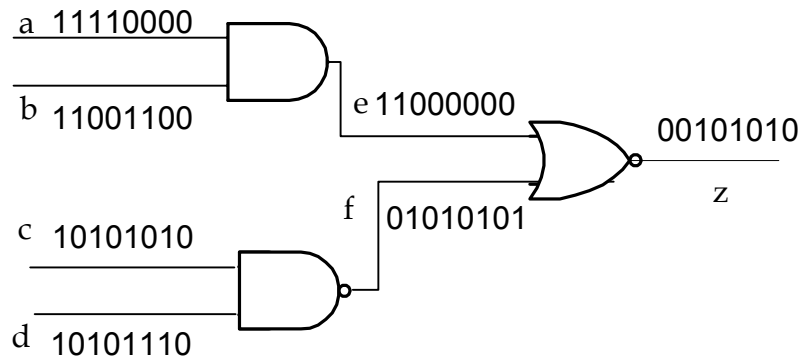


Figure 11.P5

- 11.5.** For the circuit shown in Fig. P11.5, use the stuck-at test given and find the signature of the good circuit if the time compaction used (a) Parity testing, (b) One count, and (c) Transition count. For each scheme, list the undetected faults and explain why they were not detected.

- 11.6.** Using a single-input signature analyzer (SSA), we have shown that the aliasing probability is given by

$$p(\text{aliasing}) = [2^{L-N} - 1] / [2^L - 1],$$

where  $L$  is the test of length and  $N$  is the degree of the compactor primitive characteristic polynomial  $P(X)$ .

- 11.7.** Develop the aliasing probability if instead of an SSA we use an  $N$ -bit MISR with an  $N$ -output CUT.
- 11.8.** Suppose that we record the signature every  $N$  clock cycles.
- (a) How many signatures will be formed for the test?
  - (b) What will be the aliasing probability?
  - (c) What are the advantages and disadvantages of such a scheme?