Module

8

Testing of Embedded System

Lesson

40

Built-In-Self-Test (BIST) for Embedded Systems

Instructional Objectives

After going through this lesson the student would be able to

- Explain the meaning of the term ‘Built-in Self-Test (BIST)’
- Identify the main components of BIST functionality
- Describe the various methods of test pattern generation for designing embedded systems with BIST
- Define what is a Signature Analysis Register and describe some methods to designing such units
- Explain what is a Built-in Logic Block Observer (BILBO) and describe how to use this block for designing BIST

Built-In-Self-Test (BIST) for Embedded Systems

1. Introduction

BIST is a design-for-testability technique that places the testing functions physically with the circuit under test (CUT), as illustrated in Figure 40.1 [1]. The basic BIST architecture requires the addition of three hardware blocks to a digital circuit: a test pattern generator, a response analyzer, and a test controller. The *test pattern generator* generates the test patterns for the CUT. Examples of pattern generators are a ROM with stored patterns, a counter, and a linear feedback shift register (LFSR). A typical response analyzer is a comparator with stored responses or an LFSR used as a signature analyzer. It compacts and analyzes the test responses to determine correctness of the CUT. A test control block is necessary to activate the test and analyze the responses. However, in general, several test-related functions can be executed through a test controller circuit.

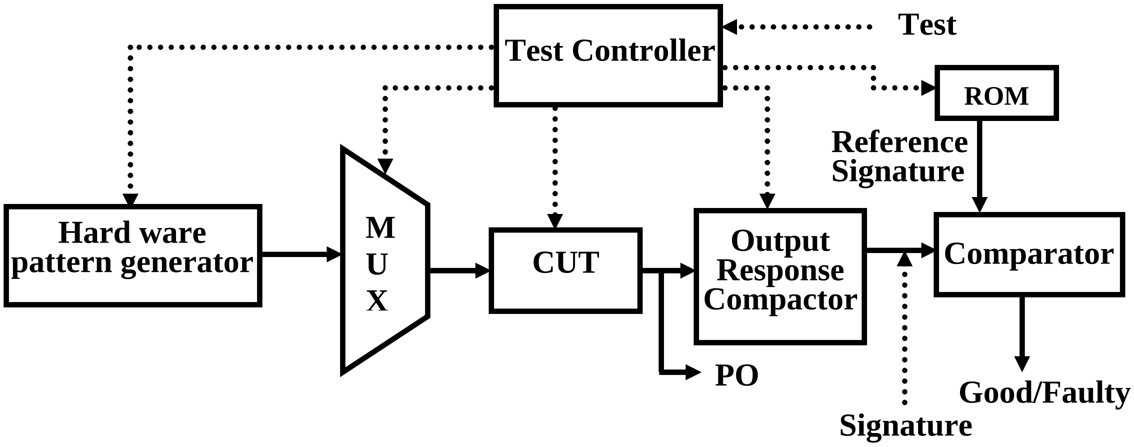


Fig. 40.1 A Typical BIST Architecture

As shown in Figure 40.1, the wires from primary inputs (PIs) to MUX and wires from circuit output to primary outputs (POs) cannot be tested by BIST. In normal operation, the CUT receives its inputs from other modules and performs the function for which it was designed. During test mode, a test pattern generator circuit applies a sequence of test patterns to the CUT, and the test responses are evaluated by a output response compactor. In the most common type of BIST, test responses are compacted in output response compactor to form (fault) *signatures*. The response signatures are compared with reference golden signatures generated or stored on-chip, and the error signal indicates whether chip is good or faulty.

Four primary parameters must be considered in developing a BIST methodology for embedded systems; these correspond with the design parameters for on-line testing techniques discussed in earlier chapter [2].

- *Fault coverage*: This is the fraction of faults of interest that can be exposed by the test patterns produced by pattern generator and detected by output response monitor. In presence of input bit stream errors there is a chance that the computed signature matches the golden signature, and the circuit is reported as fault free. This undesirable property is called *masking* or *aliasing*.
- *Test set size*: This is the number of test patterns produced by the test generator, and is closely linked to fault coverage: generally, large test sets imply high fault coverage.
- *Hardware overhead*: The extra hardware required for BIST is considered to be overhead. In most embedded systems, high hardware overhead is not acceptable.
- *Performance overhead*: This refers to the impact of BIST hardware on normal circuit performance such as its worst-case (critical) path delays. Overhead of this type is sometimes more important than hardware overhead.

Issues for BIST

- Area Overhead: Additional active area due to test controller, pattern generator, response evaluator and testing of BIST hardware.
- Pin Overhead: At least 1 additional pin is needed to activate BIST operation. Input MUX adds extra pin overheads.
- Performance overhead: Extra path delays are added due to BIST.
- Yield loss increases due to increased chip area.
- Design effort and time increases due to design BIST.
- The BIST hardware complexity increases when the BIST hardware is made testable.

Benefits of BIST

- It reduces testing and maintenance cost, as it requires simpler and less expensive ATE.
- BIST significantly reduces cost of automatic test pattern generation (ATPG).
- It reduces storage and maintenance of test patterns.
- It can test many units in parallel.
- It takes shorter test application times.
- It can test at functional system speed.

BIST can be used for non-concurrent, on-line testing of the logic and memory parts of a system [2]. It can readily be configured for event-triggered testing, in which case, the BIST control can be tied to the system reset so that testing occurs during system start-up or shutdown. BIST can also be designed for periodic testing with low fault latency. This requires incorporating a testing process into the CUT that guarantees the detection of all target faults within a fixed time.

On-line BIST is usually implemented with the twin goals of complete fault coverage and low fault latency. Hence, the test generation (TG) and response monitor (RM) are generally designed to guarantee coverage of specific fault models, minimum hardware overhead, and reasonable set size. These goals are met by different techniques in different parts of the system.

TG and RM are often implemented by simple, counter-like circuits, especially linear-feedback shift registers (LFSRs) [3]. The LFSR is simply a shift register formed from standard flip-flops, with the outputs of selected flip-flops being fed back (modulo-2) to the shift register's inputs. When used as a TG, an LFSR is set to cycle rapidly through a large number of its states. These states, whose choice and order depend on the design parameters of the LFSR, define the test patterns. In this mode of operation, an LFSR is seen as a source of (pseudo) *random* tests that are, in principle, applicable to any fault and circuit types. An LFSR can also serve as an RM by counting (in a special sense) the responses produced by the tests. An LFSR RM's final contents after applying a sequence of test responses forms a fault signature, which can be compared to a known or generated good signature, to see if a fault is present. Ensuring that the fault coverage is sufficiently high and the number of tests is sufficiently low are the main problems with random BIST methods. Two general approaches have been proposed to preserve the cost advantages of LFSRs while making the generated test sequence much shorter. Test points can be inserted in the CUT to improve controllability and observability; however, they can also result in performance loss. Alternatively, some determinism can be introduced into the generated test sequence, for example, by inserting specific "seed" tests that are known to detect hard faults.

A typical BIST architecture using LFSR is shown in Figure 40.2 [4]. Since the output patterns of the LFSR are time-shifted and repeated, they become correlated; this reduces the effectiveness of the fault detection. Therefore a phase shifter (a network of XOR gates) is often used to decorrelate the output patterns of the LFSR. The response of the CUT is usually compacted by a multiple input shift register (MISR) to a small signature, which is compared with a known fault-free signature to determine whether the CUT is faulty.

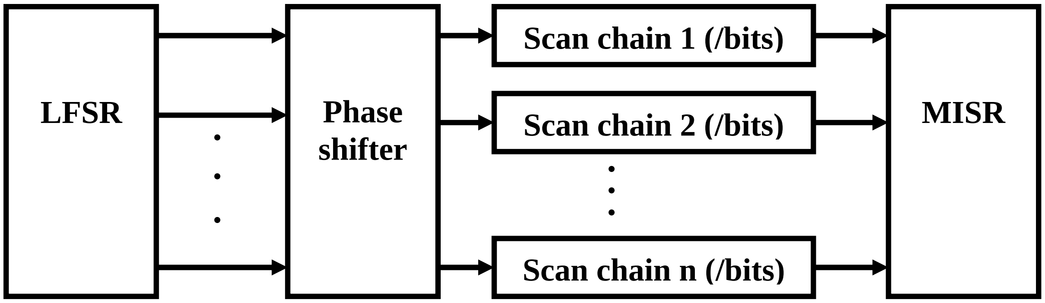


Fig. 40.2 A generic BIST architecture based on an LFSR, an MISR, and a phase shifter

2. BIST Test Pattern Generation Techniques

2.1 Stored patterns

An automatic test pattern generation (ATPG) and fault simulation technique is used to generate the test patterns. A good test pattern set is stored in a ROM on the chip. When BIST is activated, test patterns are applied to the CUT and the responses are compared with the corresponding stored patterns. Although stored-pattern BIST can provide excellent fault coverage, it has limited applicability due to its high area overhead.

2.2 Exhaustive patterns

Exhaustive pattern BIST eliminates the test generation process and has very high fault coverage. To test an n-input block of combinational logic, it applies all possible 2^n-input patterns to the block. Even with high clock speeds, the time required to apply the patterns may make exhaustive pattern BIST impractical for a circuit with n>20.

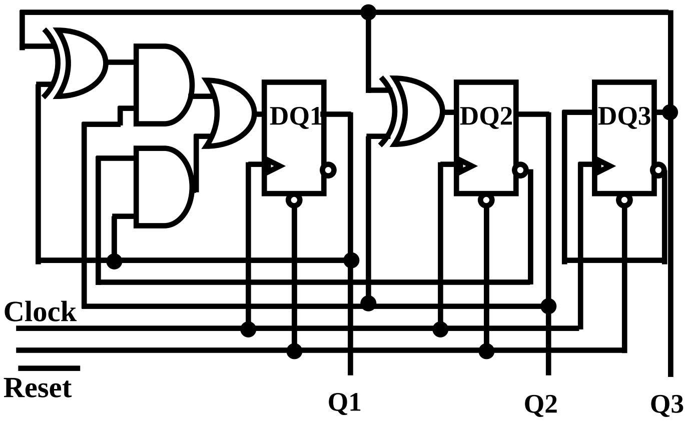


Fig. 40.3 Exhaustive pattern generator

2.3 Pseudo-exhaustive patterns

In pseudo-exhaustive pattern generation, the circuit is partitioned into several smaller sub-circuits based on the output cones of influence, possibly overlapping blocks with fewer than *n* inputs. Then all possible test patterns are exhaustively applied to each sub-circuit. The main goal of pseudo-exhaustive test is to obtain the same fault coverage as the exhaustive testing and, at the same time, minimize the testing time. Since close to 100% fault coverage is guaranteed, there is no need for fault simulation for exhaustive testing and pseudo-exhaustive testing. However, such a method requires extra design effort to partition the circuits into pseudo-exhaustive testable sub-circuits. Moreover, the delivery of test patterns and test responses is also a major consideration. The added hardware may also increase the overhead and decrease the performance.

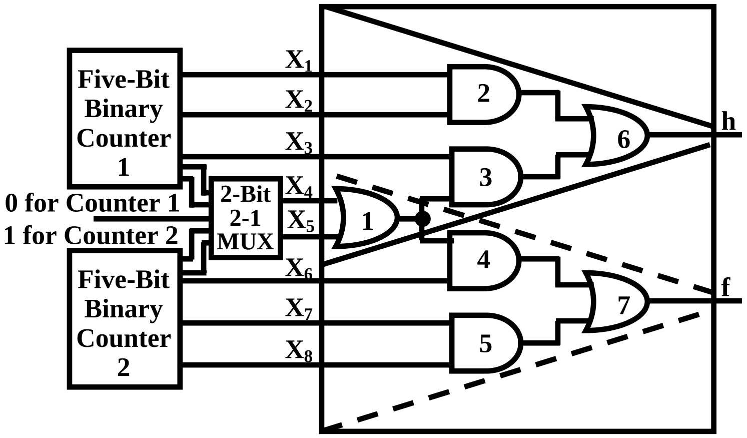


Fig. 40.4 Pseudo-exhaustive pattern generator

Circuit partitioning for pseudo-exhaustive pattern generation can be done by *cone segmentation* as shown in Figure 40.4. Here, a cone is defined as the fan-ins of an output pin. If the size of the largest cone in K, the patterns must have the property to guarantee that the patterns applied to any K inputs must contain all possible combinations. In Figure 40.4, the total circuit is divided into two cones based on the cones of influence. For cone 1 the PO *h* is influenced by X_1, X_2, X_3, X_4 and X_5 while PO *f* is influenced by inputs X_4, X_5, X_6, X_7 and X_8. Therefore the total test pattern needed for exhaustive testing of cone 1 and cone 2 is $(2^5 + 2^5) = 64$. But the original circuit with 8 inputs requires $2^8 = 256$ test patterns exhaustive test.

2.4 Pseudo-Random Pattern Generation

A string of 0's and 1's is called a pseudo-random binary sequence when the bits appear to be random in the local sense, but they are in someway repeatable. The linear feedback shift register (LFSR) pattern generator is most commonly used for pseudo-random pattern generation. In general, this requires more patterns than deterministic ATPG, but less than the exhaustive test. In contrast with other methods, pseudo-random pattern BIST may require a long test time and necessitate evaluation of fault coverage by fault simulation. This pattern type, however, has the potential for lower hardware and performance overheads and less design effort than the preceding methods. In pseudorandom test patterns, each bit has an approximately equal probability of being a 0 or a 1. The number of patterns applied is typically of the order of 10^3 to 10^7 and is related to the circuit's testability and the fault coverage required.

Linear feedback shift register reseeding [5] is an example of a BIST technique that is based on controlling the LFSR state. LFSR reseeding may be static, that is LFSR stops generating patterns while loading seeds, or dynamic, that is, test generation and seed loading can proceed simultaneously. The length of the seed can be either equal to the size of the LFSR (full reseeding) or less than the LFSR (partial reseeding). In [5], a dynamic reseeding technique that allows partial reseeding is proposed to encode test vectors. A set of linear equations is solved to obtain the seeds, and test vectors are ordered to facilitate the solution of this set of linear equations.

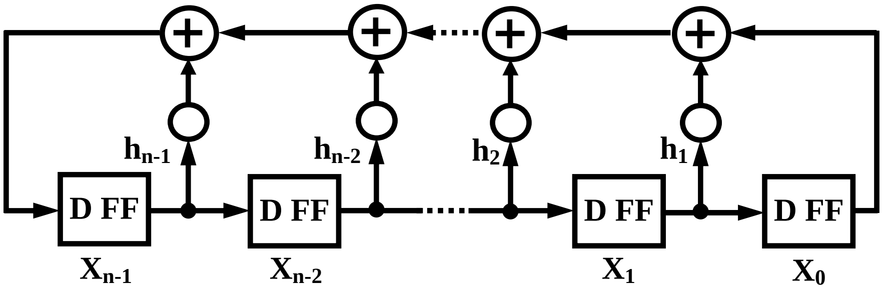


Fig. 40.5 Standard Linear Feedback Shift Register

Figure 40.5 shows a standard, external exclusive-OR linear feedback shift register. There are n flip-flops (Xn-1,……X0) and this is called n-stage LFSR. It can be a near-exhaustive test pattern generator as it cycles through 2^n-1 states excluding all 0 states. This is known as a maximal length LFSR. Figure 40.6 shows the implementation of a n-stage LFSR with actual digital circuit. [1]

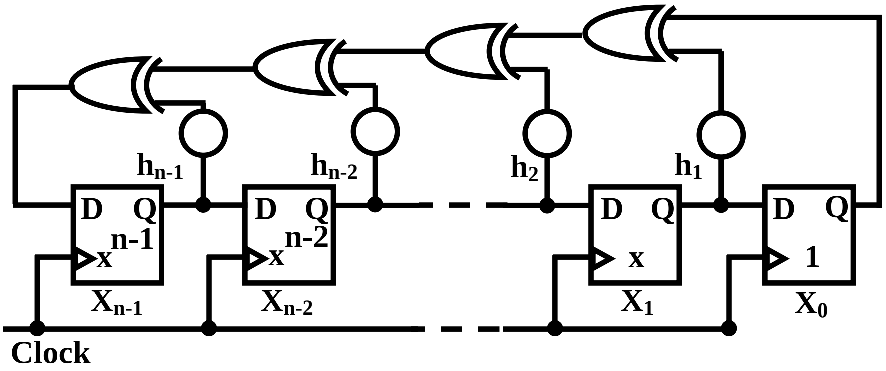


Fig. 40.6 n-stage LFSR implementation with actual digital circuit

2.5 Pattern Generation by Counter

In a BIST pattern generator based on a folding counter, the properties of the folding counter are exploited to find the seeds needed to cover the given set of deterministic patterns. Width compression is combined with reseeding to reduce the hardware overhead. In a two-dimensional test data compression technique an LFSR and a folding counter are combined for scan-based BIST. LFSR reseeding is used to reduce the number of bits to be stored for each pattern (horizontal compression) and folding counter reseeding is used to reduce the number of patterns (vertical compression).

2.6 Weighted Pseudo-random Pattern Generation

Bit-flipping [9], bit-fixing, and weighted random BIST [1,8] are example of techniques that rely on altering the patterns generated by LFSR to embed deterministic test cubes. A hybrid between pseudorandom and stored-pattern BIST, weighted pseudorandom pattern BIST is effective for dealing with hard-to-detect faults. In a pseudorandom test, each input bit has a probability of 1/2 of being either a 0 or a 1. In a weighted pseudorandom test, the probabilities, or input weights, can differ. The essence of weighted pseudorandom testing is to bias the probabilities of the input bits so that the tests needed for hard-to-detect faults are more likely to occur. One approach uses software that determines a single or multiple weight set based on a probabilistic analysis of the hard-to detect faults. Another approach uses a heuristic-based initial weight set followed by additional weight sets produced with the help of an ATPG system. The weights are either realized by logic or stored in on-chip ROM. With these techniques, researchers obtained fault coverage over 98% for 10 designs, which is the same as the coverage of deterministic test vectors.

In hybrid BIST method based on weighted pseudorandom testing, a weight of 0, 1, or μ (unbiased) is assigned to each scan chain in CUT. The weight sets are compressed and stored on the tester. During test application, an on-chip lookup table is used to decompress the data from the tester and generate weight sets. In order to reduce the hardware overhead, scan cells are carefully reordered and a special ATPG approach is used to generate suitable test cubes.

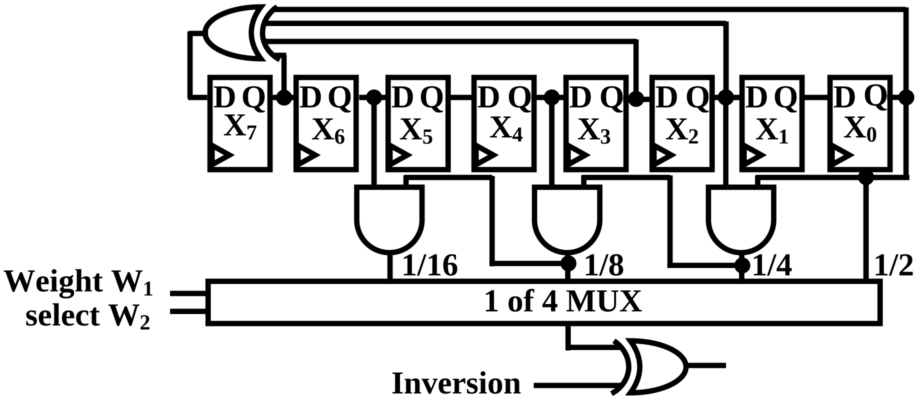


Fig. 40.7 Weighted pseudo-random pattern generator

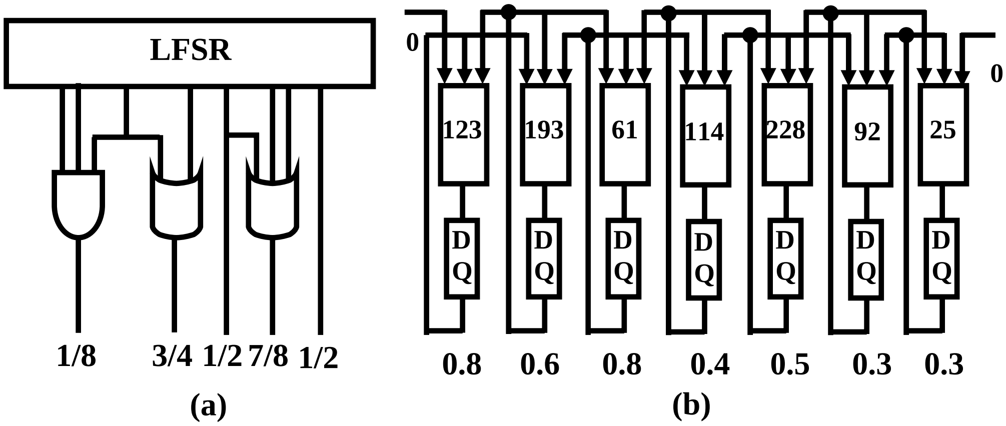
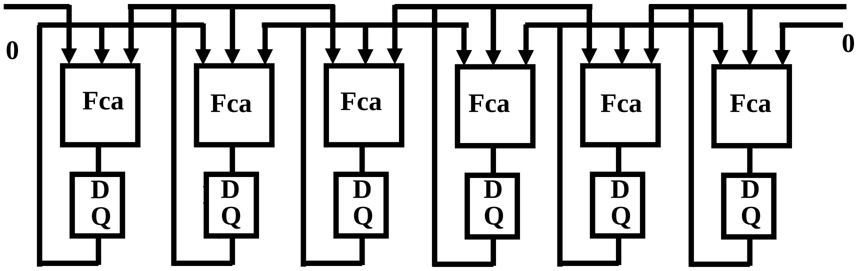


Fig. 40.8 weighted pseudorandom patterns.

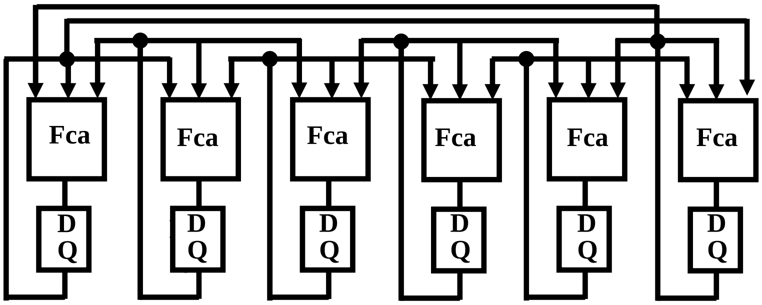
Figure 40.7 shows a weighted pseudo-random pattern generator implemented with programmable probabilities of generating zeros and ones at the PIs. As we know, LFSR generates pattern with equal probability of 1s and 0s. As shown in Figure 40.8 (a), if a 3-input AND gate is used, the probability of 1s becomes 0.125. If a 2-input OR gate is used, the probability becomes 0.75. Second, one can use cellular automata to produce patterns of desired weights as shown in Figure 40.8(b).

2.7 Cellular Automata for Pattern Generation

Cellular automata are excellent for pattern generation, because they have a better randomness distribution than LFSRs. There is no shift induced bit value correlation. A cellular automaton is a collection of cells with regular connections. Each pattern generator cell has few logic gates, a flip-flop and is connected only to its local neighbors. If C_i is the state of the current CA cell, C_{i+1} and C_{i-1} are the states of its neighboring cells. The next state of cell C_i is determined by (C_{i-1}, C_i , and C_{i+1}). The cell is replicated to produce cellular automaton. The two commonly used CA structures are shown in Figure 40.9.



(a) CA with null boundary conditions



(b) CA with null cyclic boundary conditions

Fig. 40.9 The structure of cellular automata

In addition to an LFSR, a straightforward way to compress the test response data and produce a fault signature is to use an FSM or an accumulator. However, the FSM hardware overhead and accumulator aliasing are difficult parameters to control. Keeping the hardware overhead acceptably low and reducing aliasing are the main difficulty in RM design.

2.9 Comparison of Test Generation Strategies

Implementing a BIST strategy, the main issues are fault coverage, hardware overhead, test time overhead, and design effort. These four issues have very complicated relationship. Table 1 summarizes the characteristics of the test strategies mentioned earlier based on the four issues.

Table 7.1 Comparison of different test strategies

Test Generation Methodology	***Fault Coverage***	***Hardware Overhead***	***Test Time Overhead***	***Design Effort***
Stored Pattern	High	High	Short	Large
Exhaustive	High	Low	Long	Small
Pseudo-exhaustive	High	High	Medium	Large
Pseudo-random	Low	Low	Long	Small
Weighted Pseudo-random	Medium	Medium	Long	Medium

3. BIST Response Compression/Compaction Techniques

During BIST, large amount of data in CUT responses are applied to Response Monitor (RM). For example, if we consider a circuit of 200 outputs and if we want to generate 5 million random patterns, then the CUT response to RM will be 1 billion bits. This is not manageable in practice. So it is necessary to compact this enormous amount of circuit responses to a manageable size that can be stored on the chip. The response analyzer compresses a very long test response into a single word. Such a word is called a signature. The signature is then compared with the prestored golden signature obtained from the fault-free responses using the same compression mechanism. If the signature matches the golden copy, the CUT is regarded fault-free. Otherwise, it is faulty. There are different response analysis methods such as ones count, transition count, syndrome count, and signature analysis.

Compression: A reversible process used to reduce the size of the response. It is difficult in hard ware.

Compaction: An irreversible (lossy) process used to reduce the size of the response.

a) Parity compression: It computes the parity of a bit stream.
b) Syndrome: It counts the number of 1's in the bit stream.
c) Transition count: It counts the number of times 0→1 and 1→0 condition occur in the bit stream.
d) Cyclic Redundancy Check (CRC): It is also called signature. It computes CRC check word on the bit stream.

Signature analysis – Compact good machine response into *good machine signature*. Actual signature generated during testing, and compared with good machine signature.

Aliasing: Compression is like a function that maps a large input space (the response) into a small output space (signature). It is a many-to-one mapping. Errors may occur in the in the input bit stream. Therefore, a faulty response may have the signature that matches the to the golden signature and the circuit is reported as the fault-free one. Such a situation is referred as the aliasing or masking. The aliasing probability is the possibility that a faulty response is treated as fault-free. It is defined as follows:

Let us assume that the possible input patterns are uniformly distributed over the possible mapped signature values. There are 2^m input patterns, 2^r signatures and 2^{n-r} input patterns map into given signature. Then the aliasing or masking probability

$$P(M)=\frac{\text{Number of erroneos input that map into the golden signature}}{\text{Number of faulty input responses}}$$

$$=\frac{2^{m-r}-1}{2^m-1}$$

$$\approx\frac{2^{m-r}}{2^m}\text{ for large } m$$

$$=\frac{1}{2^r}$$

The aliasing probability is the major considerations in response analysis. Due to the n-to-1 mapping property of the compression, it is unlikely to do diagnosis after compression. Therefore, the diagnosis resoluation is very poor after compression. In addition to the aliasing probability, hardware overhead and hardware compatibility are also important issues. Here, hardware compatibility is referred to how well the BIST hardware can be incorporated in the CUT or DFT.

3.1 Ones Count

The number of ones in the CUT output response is counted. In this method the number of ones is the signature. It requires a simple counter to accomplish the goal. Figure 40.10 shows the test structure of ones count for a single output CUT. For multiple output ones, a counter for each output or one output at a time with the same input sequence can be used. Input test sequence can be permuted without changing the count.

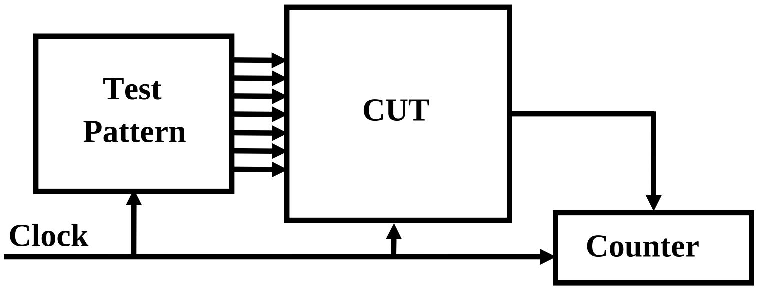


Fig. 40.10 Ones count compression circuit structure

For N-bit test length with r ones the masking probability is shown as follows:

Number of masking sequences = $\binom{N}{r}-1$

2^N possible output sequences with only one fault free.

The masking probabilities: $P(M)=\dfrac{\binom{N}{r}}{\left(2^N-1\right)}\cong(\pi N)^{-1/2}$

It has low masking probability for very small and very large r. It always detects odd number of errors and it may detect even number of errors.

3.2 Transition Count

It is very similar to ones count technique. In this method the number of transitions in the CUT response, zero to one and/or one to zero is counted. Figure 40.11 shows a test structure of transition counting. It has simple hardware DFF with EXOR to detect a transition and counter to count number of transitions. It has less aliasing probability than ones counting. Test sequences cannot be permuted. Permutation of input sequences will change the number of transitions. On the other hand, one can reorder the test sequence to maximize or minimize the transitions, hence, minimize the aliasing probability.

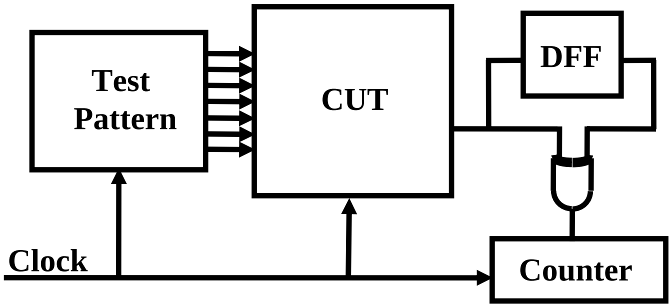


Fig. 40.11 Transition count compression circuit structure

For N-bit test length with r transitions the masking probability is shown as follows:

For the test length of N, there are *N-1* transitions.

Number of masking sequences = $\binom{N}{r}-1$

Hence, $\binom{N-1}{r}$ is the number of sequences that has r transitions.

Since the first output can be either one or zero, therefore, the total number must be multiplied by 2. Therefore total number of sequences with same transition counts : $2\binom{N-1}{r}$. Again, only one of them is fault-free.

Masking probabilities: $P(M)=\dfrac{2\binom{N-1}{2}-1}{\left(2^N-1\right)}\cong(\pi N)^{-1/2}$

3.3 Syndrome Testing

Syndrome is defined as the probability of ones of the CUT output response. The syndrome is *1/8* for a 3-input AND gate and *7/8* for a 3-input OR gate if the inputs has equal probability of ones and zeros. Figure 40.12 shows a BIST circuit structure for the syndrome count. It is very similar to ones count and transition count. The difference is that the final count is divided by the number of patterns being applied. The most distinguished feature of syndrome testing is that the syndrome is independent of the implementation. It is solely determined by its function of the circuit.

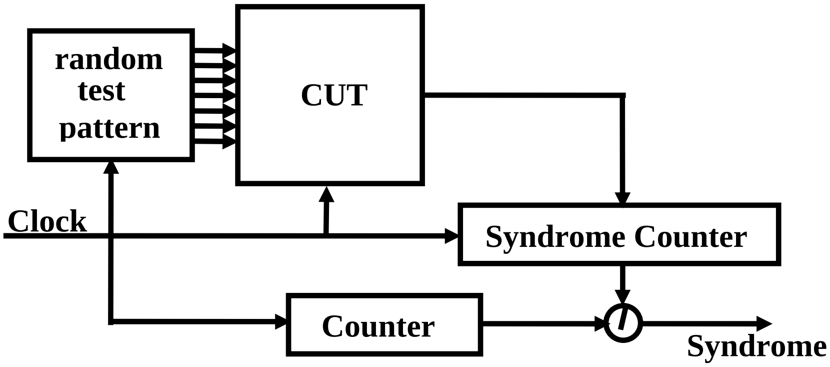


Fig. 40.12 Syndrome testing circuit structure

The originally design of syndrome test applies exhaustive patterns. Hence, the syndrome is $S = K/2^n$, where n is the number of inputs and K is the number of minterms. A circuit is syndrome testable if all single stuck-at faults are syndrome detectable. The interesting part of syndrome testing is that any function can be designed as being syndrome testable.

3.4 LFSR Structure

- External and internal type LFSR is used. Both types use D type flip-flop and exclusive-OR logic as shown in Figure 40.13.
- In *external type LFSR*, XOR gates are placed outside the shift path. It is also called *type 1 LFSR* [1].
- In *internal type LFSRs*, also called *type 2 LFSR*, XOR gates are placed in between the flip-flops.

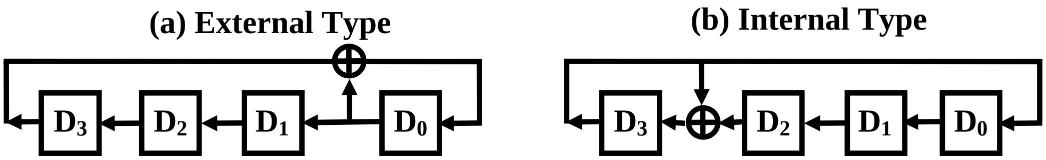


Fig. 40.13 Two types of LFSR

One of the most important properties of LFSRs is their recurrence relationship. The recurrence relation guarantees that the states of a LFSR are repeated in a certain order. For a given sequence of numbers $a_0, a_1, a_2, \ldots\ldots\ldots a_n, \ldots\ldots$ We can define a generating function:

$G(x) = a_0 + a_1x + a_2x^2 + \ldots\ldots\ldots + a_mx^m + \ldots\ldots$

$$= \sum_{m=0}^{\alpha} a_m x^m$$

$$\{a_m\} = \{a_0, a_1, a_2, \ldots\ldots\}$$

where $a_i = 1 \text{ or } 0$ depending on the out put stage and time t_i.

The initial states are $a_{-n}, a_{-n+1}, \ldots\ldots, a_{-2}, a_{-1}$. The recurrent relation defining $\{a_m\}$ is

$$a_m = \sum_{i=1}^{n} c_i a_{m-i}$$

where $c_i = 0$, means output is not fed back

$= 1$, otherwise

$$G(x) = \sum_{m=0}^{\alpha}\sum_{i=1}^{n} c_i a_{m-i} x^m$$

$$= \sum_{i=1}^{n} c_i x^i \sum_{m=0}^{\alpha} a_{m-i} x^m$$

$$= \sum_{i=1}^{n} c_i x^i \left[a_{-i} x^i + \ldots. + a_{-1} x^{-1} + \sum_{m=0}^{\alpha} a_m x^m \right]$$

$$G(x) = \frac{\sum_{i=1}^{n} c_i x^i \left(a_{-i} x^i + \ldots. + a_{-1} x^{-1} \right)}{1 - \sum_{i=1}^{n} c_i x^i}$$

G(x) has been expressed in terms of the initial state and the feedback coefficients. The denominator of the polynomial G(x), $f(x) = 1 - \sum_{i=1}^{n} c_i x^i$ is called the characteristic polynomial of the LFSR.

3.5 LFSR for Response Compaction: Signature Analysis

- It uses *cyclic redundancy check code* (CRCC) generator (LFSR) for response compacter
- In this method, data bits from circuit Pos to be compacted as a decreasing order coefficient polynomial
- CRCC divides the PO polynomial by its characteristic polynomial that leaves remainder of division in LFSR. LFSR must be initialized to *seed value* (usually 0) before testing.
- After testing, signature in LFSR is compared to known good machine signature

For an output sequence of length N, there is a total of 2^N-1 faulty sequence. Let the input sequence is represented as *P(x)* as *P(x)=Q(X)G(x)+R(x)*. *G(x)* is the characteristic polynomial; *Q(x)* is the quotient; and *R(x)* is the remainder or signature. For those aliasing faulty sequence, the remainder *R(x)* will be the same as the fault-free one. Since, *P(x)* is of order N and *G(x)* is of order *r*, hence *Q(x)* has an order of *N-r*. Hence, there are 2^{N-r} possible *Q(x)* or *P(x)*. One of them is fault-free. Therefore, the aliasing probability is shown as follows: $P(M) = \frac{2^{N-r}-1}{2^N-1} \cong 2^{-r}$ for large N. Masking probabilities is independent of input sequence.

Figure 40.14 illustrates a modular LFSR as a response compactor.

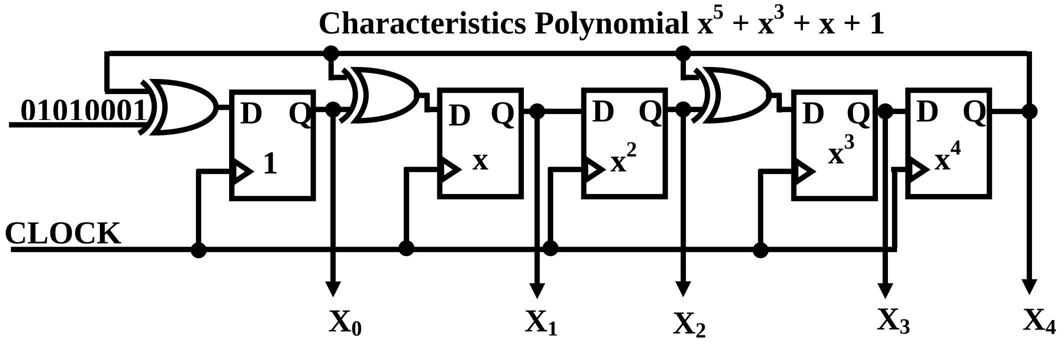


Fig. 40.14 Modular LFSR as a response compactor

- Any divisor polynomial G(x) with two or more non-zero coefficients will detect all single-bit errors.

3.6 Multiple-Input Signature Register (MISR)

- The problem with ordinary LFSR response compacter is too much hardware overhead if one of these is put on each *primary output* (PO).
- Multiole-input signature register (MISR) is the solution that compacts all outputs into one LFSR. It works because LFSR is linear and obeys *superposition principle*.
- All responses are superimposed in one LFSR. The final remainder is XOR sum of remainders of polynomial divisions of each PO by the characteristic polynomial.

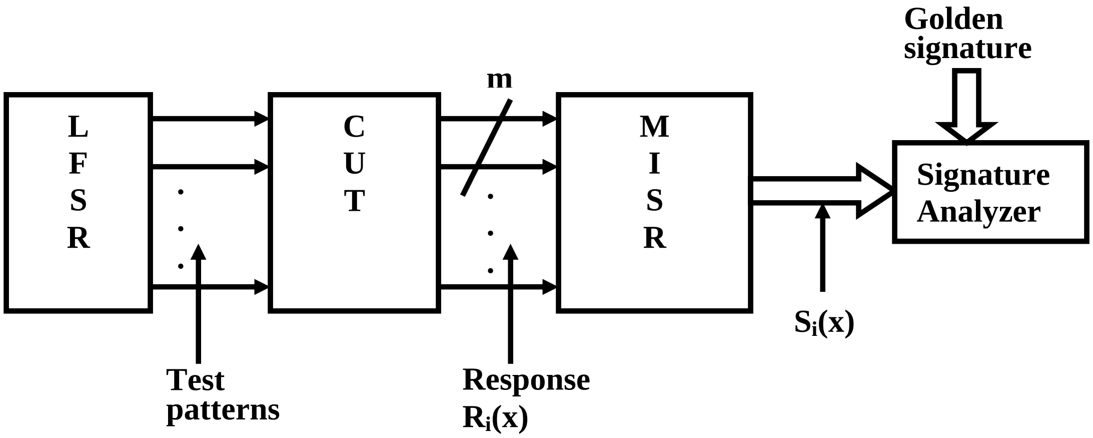


Fig. 40.15 Multiple input signature register

Figure 40.15 illustrates a m-stage MISR. After test cycle *i*, the test responses are stable on CUT outputs, but the shifting clock has not yet been applied.
Ri(x)= (m-1)th polynomial representing the test responses after test cycle *i*.
Si(x)=polynomial representing the state of the MISR after test cycle *i*.

$$R_i(x) = r_{i,m-1}x^{m-1} + r_{i,m-2}x^{m-2} + \dots\dots + r_{i,1}x + r_{i,0}$$

$$S_i(x) = S_{i,m-1}x^{m-1} + S_{i,m-2}x^{m-2} + \dots\dots + S_{i,1}x + S_{i,0}$$

$$S_{i+1}(x) = \left[R_i(x) + xS_i(x)\right] \bmod G(x)$$

$G(x)$ is the characteristic polynomial

Assume initial state of MISR is 0. So,

$$S_0(x) = 0$$

$$S_1(x) = \left[R_0(x) + xS_0(x)\right] \bmod G(x) = R_0(x)$$

$$S_2(x) = \left[R_1(x) + xS_1(x)\right] \bmod G(x) = \left[R_1(x) + R_0(x)\right] \bmod G(x)$$

.

.

$$S_n(x) = \left[x^{n-1}R_0(x) + x^{n-2}R_1(x) + \dots\dots + xR_{n-2}(x) + R_{n-1}(x)\right] \bmod G(x)$$

This is the signature left in MISR after *n* patterns are applied. Let us consider a n-bit response compactor with m-bit error polynomial. Then the error polynomial is of (m+n-2) degree that gives (2^{m+n-1}-1) non-zero values. G(x) has 2^{n-1}-1 nonzero multiples that result m polynomials of degree <=m+n-2.

Probability of masking
$$P(M) = \frac{2^{n-1}-1}{2^{m+n-1}-1}$$
$$\approx \frac{1}{2^m}$$

3.7 Logic BIST Architecture

- Test-per-clock system
    - More hardware, less test time.
    - BILBO: Built in logic bloc observer
- Test-per-scan system.
    - Less hardware, more test time.
    - STUMPS: Self-Test using a MISR and Parallel Shift register.
- Circular self-test path
    - Lowest hardware, lowest fault coverage.

3.7.1 Test-Per-Clock BIST

Two different test-per-clock BIST structures are shown in Figure 40.16. For every test clock, LFSR generates a test vector and Signature Analyzer (MISR) compresses a response vector. In every clock period some new set of faults is tested. This system requires more hardware. It takes less test time. It can be used for exhaustive test, pseudo-exhaustive test, pseudorandom testing, and weight pseudorandom testing.

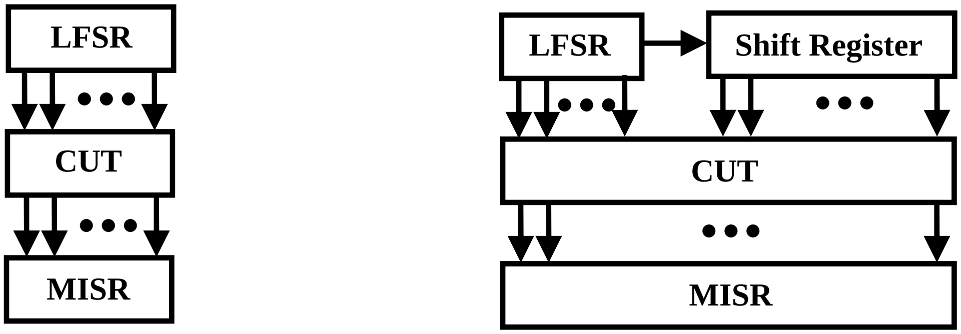


Fig. 40.16 Test-Per-Clock BIST structure

3.7.2 Built-in Logic Block Observer (BILBO)[1]

Built-in logic block observation is a well known approach for pipelined architecture. It adds some extra hardware to the existing registers (D flip-flop, *pattern generator*, *response compacter*, & *scan chain*) to make them multifunctional. All FFs are reset to 0. The circuit diagram of a BILBO module is shown in Figure 40.17. The BILBO has two control signals (B1 and B2).

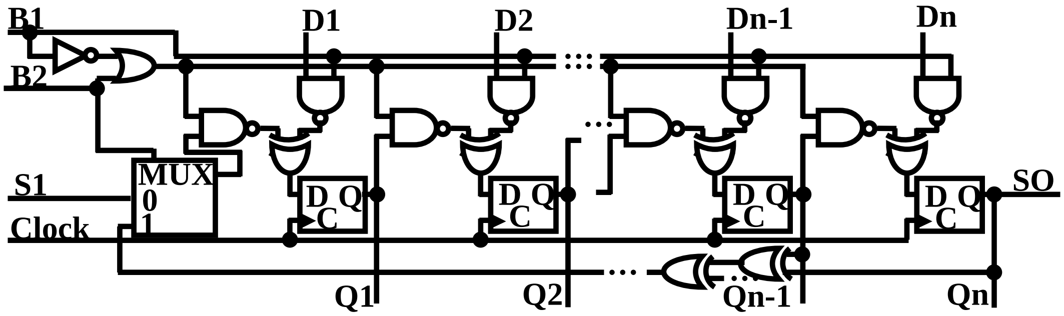


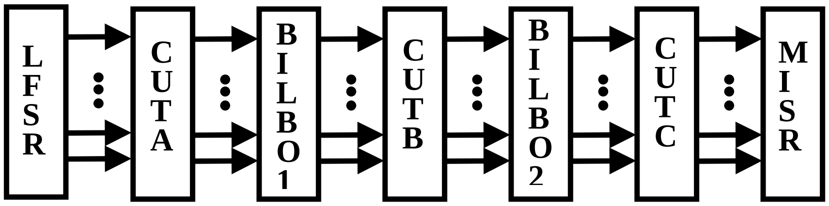
Fig. 40.17 BILBO Example

Four different modes of BILBO operation

(a) Scan-in-Scan-out: shift register
(b) Normal register mode: PIPO register
(c) Pattern generator mode: LFSR
(d) Response compactor mode: MISR

3.7.3 BILBO Usage for multi-CUT structure [1]

As shown in Figure 40.18, in this BILBO structure, multiple modules can be tested simultaneously. The total operation is done in two phase as stated below.



(a) Example test configuration.

Fig. 40.18 Circuit configured with BILBO

Phase 1

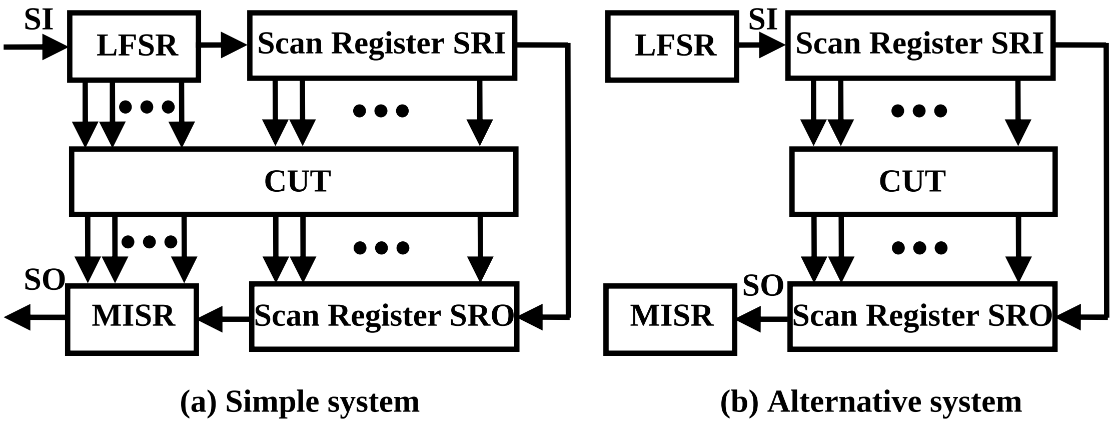
In this mode of operation BILBO1 operates in MISR mode and BILBO2 operates in LFSR mode. CUT A and CUT C are tested in parallel.

Phase 2

In this of operation BILBO1 operates in LFSR mode and BILBO2 operates in MISR mode. Only CUT B is tested in this mode of operation.

3.7.4 Test-Per-Scan BIST

Instead of using LFSR and MISR for every input/output pins, this approach combine LFSR/MISR with shift register to minimize the hardware overhead. Figure 40.19 shows the basic circuit structure of a test-per-scan BIST. In BIST mode, LFSR generates test vectors and shifted to the inputs of the CUT via scan register. At the same time, the response are scanned in and compressed by the LFSR. Due to the use of scan chain for the delivery of test patterns and responses, the test speed is much slower than the test-per-clock approach. The clocks required for a test cycle is the maximal of the scan stages of input and output scan registers. Also fall in this category include CEBS, LOCST, and STUMP.



(a) Simple system (b) Alternative system

Fig. 40.19 Basic test-per-scan structure

3.7.5 Self-Testing Using MISR and Parallel Shift register sequence generator (STUMP)

The architecture of the self-testing using MISR and parallel SRSG (STUMP) is shown in Figure 40.20. Instead of using only one scan chain, it uses multiple scan chains to minimize the test time. Since the scan chains may have different lengths, the LFSR runs for N cycles (the length of the longest scan chain) to load all the chains. For such a design, the internal type LFSR is preferred. If the external type is used, the difference between two LFSR output bits is only the time shift. Hence, the correlation between two scan chains can be very high.

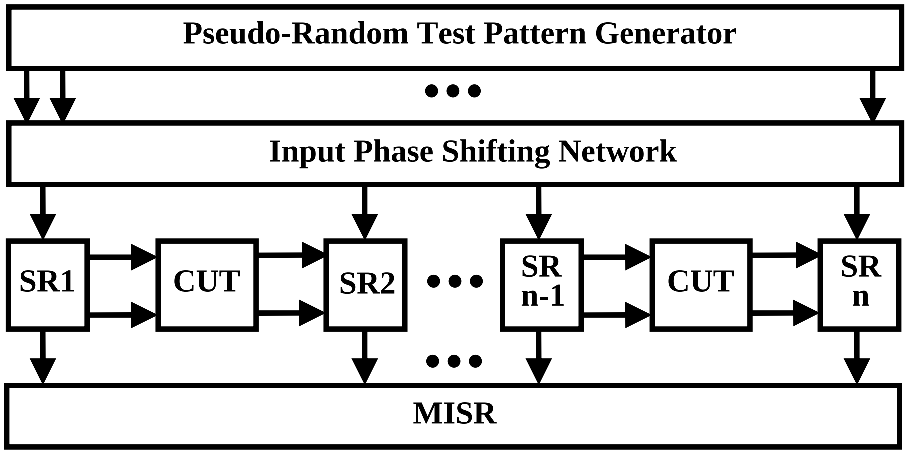

Pseudo-Random Test Pattern Generator
Input Phase Shifting Network
SR1
CUT
SR2
SR n-1
CUT
SR n
MISR

Fig. 40.20 STUMPS test-per-scan testing system

Test Procedure of STUMP

1. Scan in patterns from LFSR to all scan chain.
2. Switch to normal function mode and apply one clock.
3. Scan out chains into MISR.
4. Overlap steps 1 and 3.

4. BIST for Structured Circuits

Structured design techniques are the keys to the high integration of VLSI circuits. The structured circuits include read only memories (ROM), random access memories (RAM), programmable logic array (PLA), and many others. In this section, we would like to focus on PLAs because they are tightly coupled with the logic circuits. While, memories are usually categorized as different category. Due to the regularity of the structure and the simplicity of the design, PLAs are commonly used in digital systems. PLAs are efficient and effective for the implementation of arbitrary logic functions, combinational or sequential. Therefore, in this section, we would like to discuss the BIST for PLAs.

A PLA is conceptually a two level AND-OR structure realization of Boolean function. Figure 40.21 shows a general structure of a PLA. A PLA typically consists of three parts, input decoders, the AND plane, the OR plane, and the output buffer. The input decoders are usually implemented as single-bit decoders which produce the direct and the complement form of inputs. The AND plane is used to generate all the product terms. The OR plane sum the required product terms to form the output bits. In the physical implementation, they are implemented as NAND-NAND or NOR-NOR structure.

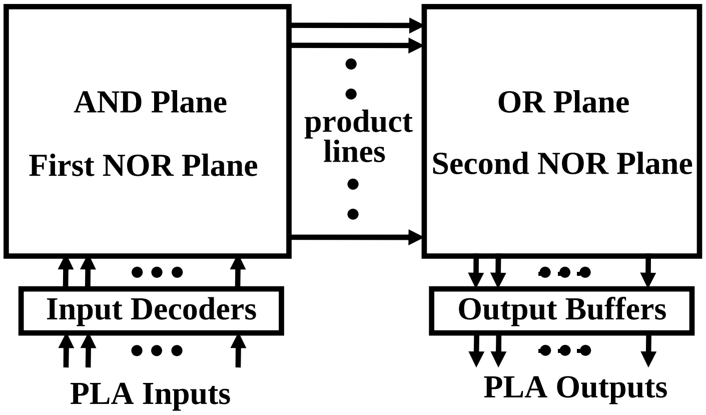


Fig. 40.21 A general structure of a PLA.

As mentioned earlier in the fault model section, PLAs has the following faults, stuck-at faults, bridging faults, and crosspoint faults. Test generation for PLAs is more difficult than that for the conventional logic. This is because that PLAs have more complicated fault models. Further, a typical PLA may have as many as 50 inputs, 67 inputs, and 190 product terms [10-11]. Functional testing of such PLAs can be a difficult task. PLAs often contain unintentional and unidentifiable redundancy which might cause fault masking. Further more, PLAs are often embedded in the logic which complicates the test application and response observation. Therefore, many people proposed the use of BIST to handle the test of PLAs.

5. BIST Applications

Manufactures are increasingly employing BIST in real products. Examples of such applications are given to illustrate the use of BIST in semiconductor, communications, and computer industrial.

5.1 Exhaustive Test in the Intel 80386 [12]

Intel 80386 has BIST logic for the exhaustive test of three control PLAs and three control ROMs. For PLAs, the exhaustive patterns are generated by LFSRs embedded in the input registers. For ROMs, the patterns are generated by the microprogram counter which is part of the normal logic. The largest PLA has 19 input bits. Hence, the test length is 512K clock cycles. The test responses are compressed by MISRs at the outputs. The contents of MISRs are continuously shifted out to an LFSR. At the end of testing, the contents of LFSRs are compared.

5.2 Pseudorandom Test in the IBM RISC/6000 [13]

The RISC/6000 has extensive BIST structure to cover the entire system. In accord with their tradition, RISC/6000 has full serial scan. Hence, the BIST it uses is the pseudorandom testing in the form of STUMPS. For embedded RAMs, it performs self-test and delay testing. For the BIST, it has a on chip processor (COP) on each chip. In COP, there are an LFSR for pattern generation, a MISR for response compression, and a counter for address counting in RAM bist. The COP counts for less than 3% of the chip area.

5.3 Embedded Cache Memories BIST of MC68060 [14]

MC68060 has two test approaches for embedded memories. First it has adhoc direct memory access for manufacturing testing because it has the only memory approach that meets all the design goals. The adhoc direct memory acess uses additional logic to make address, data in, data out, and control line for each memory accessible through package pins. An additional set of control signals selects which memory is activated. The approach makes each memory visible through the chip pins as though it is a stand-alone memory array. For the burn-in test, it builds the BIST hardware around the adhoc test logic. The two-scheme approach is used because it meets the burn-in requirements with little additional logic.

5.4 ALU Based Programmable MISR of MC68HC11 [15]

Broseghini and Lenhert implemented an ALU-Based self-test system on a MC68HC11 Family microcontroller. A fully programmable pseudorandom pattern generator and MISR are used to reduce test length and aliasing probabilities. They added microcodes to configure ALU into a LFSR or MISR. It transforms the adder into a LFSR by forcing the carry input to 0. With such a feature, the hardware overhead is minimized. The overhead is only 25% as compare to the implementation by dedicated hardware.

References

[1] M. L. Bushnell and V. D Agarwal, “Essentials of Electronic Testing” Kluwer academic Publishers, Norwell, MA, 2000.
[2] H. Al-Asaad, B. T. Murray, and J. P. Hayes, “Online BIST for embedded systems” IEEE Design & Test of Computers, Volume 15, Issue 4, Oct.-Dec. 1998 Page(s): 17 – 24
[3] M. Abramovici, M.A. Breuer, AND A.D. Friedman, “Digital Systems Testing and Testable Design”, *IEEE Press* 1990.
[4] R. Zurawski, “Embedded Systems Handbook”, *Taylor & Francis,* 2005.

[5] C. V. Krishna, A. Jalas, and N. A. Tauba, "Test vector encoding using partial LFSR reseeding", in *Proceeding of the International Test Conference*, pp. 885-893, 2001.

[6] J. Rajski, J. Tyszer, and N. Zacharia, "Test data decompression for multiple scan designs with boundary scan", *IEEE Transactions on Computers*, 47, pp. 1188-1200, 1998.

[7] N. A. Tauba and E.J.MaCluskey, "Altering a pseudo-random bit sequence for scan based", in *Proceedings of International Test Conference*, 1996, pp. 167-175.

[8] S. Wang, "Low hardware overhead scan based 3-weight weighted random BIST", in *Proceedings of International Test Conference*, 2001, pp. 868-877.

[9] H. –J. Wunderlich and G.Kiefer, "Bit-flipping BIST", in Proceedings of International Conference on Computer-Aided Design, 1996, pp. 337-343.

[10] C.Y. Liu, K.K Saluja, and J.S. Ypadhyaya, "BIST-PLA: A Built-in Self-Test Design of Large Programmable Logic Arrays," *Proc. 24th Design Automation Conf.,* June 1987, pp. 385-391.

[11] C.Y.Liu and K.K.Saluja, "Built -In Self-Test Techniques for Programmable logic Arrays," in *VLSI Fault Modeling and Testing Techniques,* G. W. Zobrist,ed., Ablex Publishing, Norwood, N.J.,1993.

[12] P. Gelsinger, "Design and Test of the 80386," *IEEE Design & Test of Computers,* Vol. 4, No. 3, June 1987, pp.42-50.

[13] I.M. Ratiu and H.B. Bakouglu, "Pseudorandom Built-In Self-Test Methodology and implementation for the IBM RISC System/6000 Processor," IBM J. Research and Development, Vol. 34. 1990, pp.78-84.

[14] A.L. Crouch, M. Pressly, J. Circello, "Testability Features of the MC68060 Microprocessor," Proc. Int'l Test Conf., 1994, pp. 60-69.

[15] J. Broseghini and D.H. Lenhert, "An ALU-Based Programmable MISR/Pseudorandom Generator for a MC68HC11 Family Self-Test," Proc. Int'l Test Conf., 1993, pp. 349-358.

Problems

1. What is Built-In-Self-Test? Discuss the issues and benefits of BIST. Describe BIST architecture and its operation.
2. Excluding the circuit under test, what are the four basic components of BIST and what function does each component perform?
3. Which two BIST components are necessary for system-level testing and why?
4. What are the different techniques for test pattern generation?
5. Discuss exhaustive and pseudo-exhaustive pattern generation. Give an example to show that pseudo-exhaustive testing requires less number of test pattern than exhaustive testing.
6. What is pseudorandom pattern generation? What is an LFSR? Describe pattern generation using LFSR.
7. Make a comparison of different test strategies based on fault coverage, hardware overhead, test time overhead and design effort.
8. An LFSR based signature register compresses an n-bit input pattern into an m-bit signature. Derive an expression for the probability of aliasing. Clearly state any assumptions you make.
9. Design a weighted pseudo-random pattern generator with programmable weights 1/2, 1/4, 11/32 and 1/16.
10. Prove that the number of 1's in an m-sequence differs from the number of 0's by one.

11. Consider a LFSR based pattern generator where the feedback network is a single XOR gate before the first stage. If the number of (feedback) inputs to the XOR is odd, is it possible for the LFSR to generate maximal length sequence? Justify or contradict.
12. Show the schematic diagram of a 4-bit BILBO register.
13. A given data path has p number of n-bit registers. For having BIST capability, suppose a% of the registers are converted to BILBO. Estimate the percentage overhead in the registers in terms of extra hardware. All gates may be assumed to have unit cost in your calculation.
14. It is said that by adding some extra hardware, a combinational circuit can be made syndrome testable for single stuck-at faults. Illustrate the process for a circuit realizing the Boolean function f = AB + B'C.
15. Define the following:
    a) Compression
    b) Compaction
    c) Signature analysis
    d) Aliasing or masking
16. Describe different response compaction techniques.
17. What are different types of LFSR? What is modular LFSR? What is characteristic polynomial?
18. Implement a standard LFSR for the characteristic polynomial $f(x) = x^8+x^7+x^2+1$.
19. Given the polynomial $P(x)=x^4+x2+x+1$:
    a. Design an external feedback LSFR with characteristic polynomial $P(x)$.
    b. Starting this LFSR in the all 1s state, determine the sequence produced.
    c. Is this a maximal length LFSR?
    d. Is the characteristic polynomial primitive?
20. Describe how LFSR is used in signature analysis for response compaction.
21. For an internal feedback Signature Analysis Register (SAR) with characteristic polynomial $P(x)=x6+x2+1$:
    a) Draw a logic diagram for the complete register.
    b) Determine the resultant signature that would be obtained for the following serial sequence of output responses produced by a known good CUT assuming the SAR is initialized to the all 0s state. Give the binary value of the resultant signature as it would be contained in the SAR in your logic diagram above.

       101001010010 ← *time*
22. What is MISR? Give architecture of an m-stage MISR and derive its signature. What is the masking probability of MISR?
23. Describe with example and diagram what are test-per-clock system and test-per-scan system. What is the difference between them?
24. What is BILBO? Describe BILBO architecture and its operation?
25. Describe how BILBO is implemented in digital circuits?
26. Describe STUMPS testing system and its test procedure.
27. Give some examples of practical BIST application in industry.