BIST FOR SYSTEMS-ON-A-CHIP

Hans-Joachim Wunderlich, University of Stuttgart

ABSTRACT

An increasing part of microelectronic systems is implemented on the basis of predesigned and preverified modules, so-called cores, which are reused in many instances. Core-providers offer RISC-kernels, embedded memories, DSPs, and many other functions, and built-in self-test is the appropriate method for testing complex systems composed of different cores.

In this paper, we overview BIST methods for different types of cores and present advanced BIST solutions. Special emphasis is put on deterministic BIST methods as they do not require any modifications of the core under test and help to protect intellectual property (IP).

1 INTRODUCTION

The recent technology developments allow embedding a large number of functional blocks into single devices and packaging the devices in very dense multi-chip modules again. The driving factors are improvements of the process technology allowing a multi-million gates fabrication and the design technology based on the reuse of intellectual property (IP). Embedded cores replace standard ICs from multiple sources and will be the predominant design style in the near future.

Cores are predesigned, preverified complex functional blocks, which are currently available as processor cores, DSP cores, memories, and as specific functions for cache controllers, interfaces, multi-media or telecommunication applications, e.g. We may classify cores based on the level of hardware description or based on the degree of integration. *Soft cores* are described at behavioral level or register transfer level, *firm cores* are gate level netlists, and *hard cores* are layouts. Based on integration we distinguish between *mergeable cores* which are designed for integration with *user defined logic (UDL)*, and non-mergeable cores designed for interaction with the UDL. Mergeable cores come as soft or firm cores while non-mergeable cores are typically hard or firm cores which remain as distinct entities sometimes in an encrypted form.

The major advantages of the System-on-a-Chip (SoC) technique are a short time to market due to the predesign, less cost due to reusability, a higher performance because of using optimized algorithms and less hardware area caused by using optimized designs. But the SoC technique also introduces new difficulties into the test process caused by the increased complexity of the chip, the reduced accessibility of the cores and the higher heterogeneity of the modules.

In the SoC test process, a core test strategy has to be determined first. We have to decouple core level testing form system test, to define an adequate core test method, and to prepare the cores for test. Then a SoC test strategy has to be selected where the test access for individual cores is determined, tests for the user-defined function are prepared, and the tests are integrated at system level. All these tasks are simplified if the cores and the entire system support a built-in self-test strategy. Equipping the cores with BIST features is preferable if the modules are not accessible externally, and it helps to protect intellectual property (IP) as less test information about the core has to be given to the user.

In this paper, we describe how user defined control logic can be synthesized so that self-test features are integrated automatically. For mergeable cores and hard cores equipped with a scan path, deterministic BIST methods only require a scan design and can be kept apart from the mission logic. Such a non-intrusive proceeding avoids timing penalties and a possible need for a redesign. Hard cores without a known structure can be tested by a functional BIST approach where the functional units are controlled in such a way that they generate precomputed deterministic test sets.

In the next section we introduce the basics and the limits of classic BIST methods. Section 3 describes how BIST is introduced into mergeable cores and user defined logic. In section 4 deterministic BIST methods are discussed.

2 BASICS AND LIMITS OF CLASSIC BIST METHODS

A self-testable module requires to incorporate a test pattern generator (TPG), a test response evaluator (TRE) and a BIST control unit (BCU). An appropriate design of the BCU allows a hierarchic BIST strategy as shown in Figure 1.

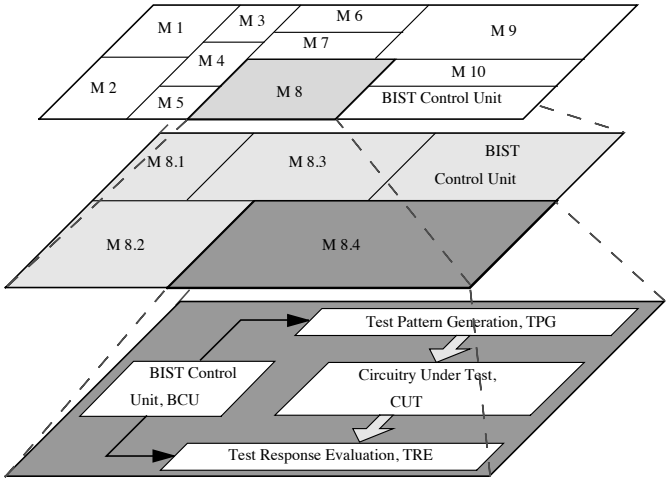


Figure 1: Hierarchic BIST

The most widespread BIST schemes for modules are the test-per-scan scheme and the test-per-clock scheme. Test-per-scan schemes use a complete or partial scan path which is serially

filled by the TPG (Figure 2) [1]. At a capture clock the content of the scan chain is applied to the module under test (MUT), and the MUT response is loaded into the scan chain in parallel. Then concurrently a new bit stream is shifted in, and the scan path output is compressed by the TRE. The test process can be accelerated if multiple scan chains are used [2].

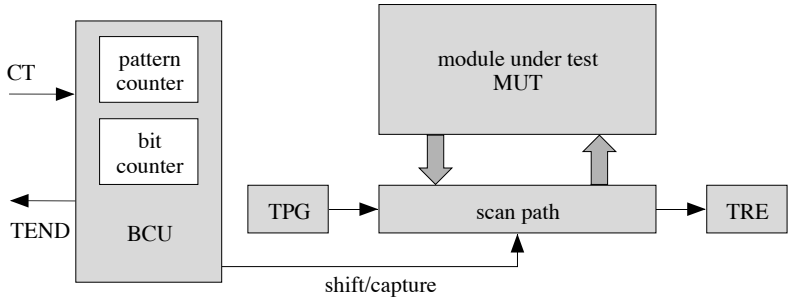


Figure 2: Test-per-scan scheme

The BIST control unit (BCU) must at least contain a bit counter for detecting, when the scan chain is filled, and a pattern counter for finalizing the test. The test-per-scan scheme fits in any commercial design flow which supports scan design, and can easily be extended to a partial scan design and multiple scan paths. The BIST hardware is mainly kept apart from the mission logic, and the performance degradation is not higher than the impact of a scan design for external testing. The BIST control unit and the overall hardware overhead are smaller than the overhead of a test-per-clock scheme. Drawbacks of the test-per-scan scheme are the long test time for serial pattern generation, and the low detectability of transition faults which require a two-pattern test.

A *test-per-clock* scheme uses special registers which work in four modes. In the *system mode* they operate just as D-type flip-flops, in the *pattern generation mode* they perform autonomous state transitions, and the states are the test patterns, in the *response evaluation mode* the responses of the MUT are compressed, and in the *shift mode* the registers work as a scan path. The first proposal of such a register was the Built-In Logic Block Observer (BILBO) by Koenemann, Mucha and Zwiehoff [3].

In the pattern generation mode, the BILBO is configured as a linear feedback shift register (LFSR). The original proposal did not distinguish between pattern generation and test response evaluation mode. Later versions re-encoded the control lines, and it has been proven advantageous to reserve one control line b_0 for switching between the *global mode* and the *local mode*. The global mode covers system mode and shift mode where all the registers perform in the same way. In the local mode the registers may work differently, some generate patterns and others evaluate responses (Figure 3).

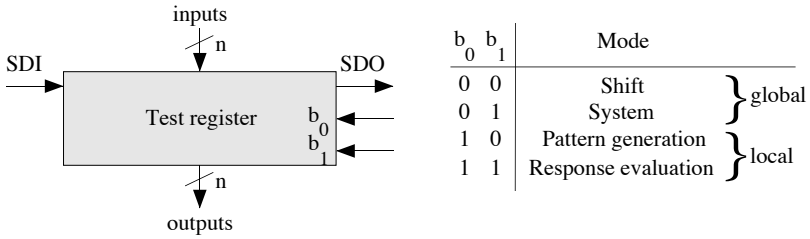


Figure 3: Control signals of a test register

The advantage of this control encoding is the fact that the BCU only needs to generate a single b_0 signal for all the registers, and only the b_1 signals must be different since a test register cannot do evaluation and pattern generation simultaneously. Hence, the test registers have to be placed in such a way that there is no direct feedback loop of a register.

In general, it is not possible to partition the flip-flops into just two sets so that there are always two corresponding test registers without self-loops. In consequence, the number of test registers must be increased, and the BIST schedule is getting more complex. A *test unit* is the minimum portion of a circuit which can be tested independently, and it consists of exactly one test register R_a for response evaluation, the circuitry under test observed by R_a and all the test registers R_j which have to generate patterns for this circuitry (Figure 4) [4,5]. A test unit is uniquely identified by the observing register R_a.

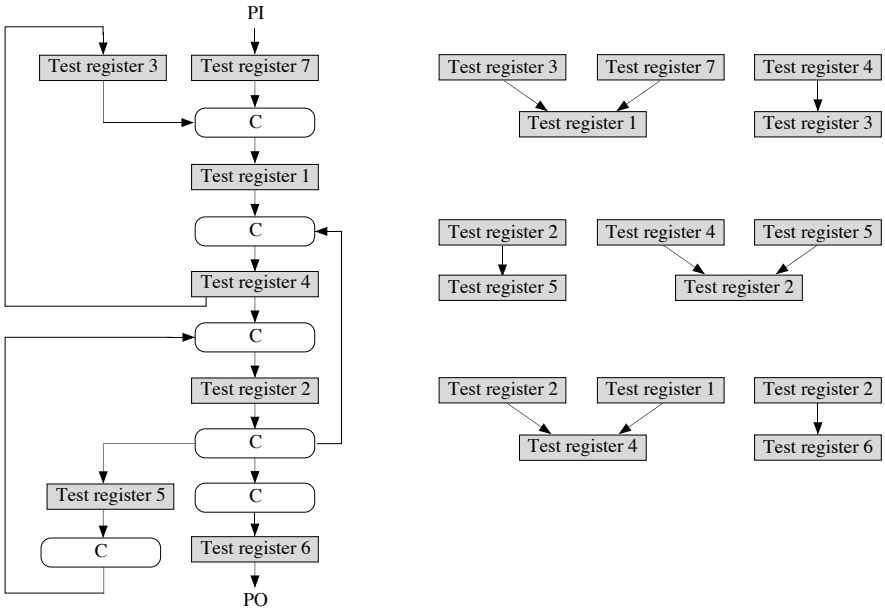


Figure 4: RT-example and test units

Two test units can be processed in parallel if there is no conflict of resources, i.e. there is no register generating patterns and evaluating responses simultaneously. Short test times require a maximum parallelism which can be obtained by solving the minimum color problem of the *test incompatibility graph* of which the nodes are test units and the edges denote a conflict between

test units. For the test units of Figure 4 we need three different colors for the three sets {TR1, TR2}, {TR3}, and {TR4, TR5, TR6}, and all test units with the same color may be tested in parallel.

The objective of an efficient BIST scheduling is not only minimizing test time but also minimizing the control effort. A *test session* is defined as a set of test units processed in parallel, and a BIST schedule is a series of test sessions which is implemented by the BCU in hardware [6,7]. The input of the BCU is at least a signal CT for starting BIST, and the outputs are a signal TEND for indicating the end of test, a global test signal TEST for controlling the b_0 inputs of the test registers, and a bundle $\vec{S}$ of local test signals b_1.

Hardware is reduced by minimizing the number of signals $\vec{S}$ which corresponds to coloring the *control incompatibility graph*. Here, the nodes are test registers, and an edge denotes the fact that one register generates patterns and the other evaluates responses during the same test session. For the circuit of Figure 4, only three different control signals are required and the complete BIST structure looks like Figure 5. The BCU has to contain a pattern counter, the number of patterns for each test session, and the assignment for each session.

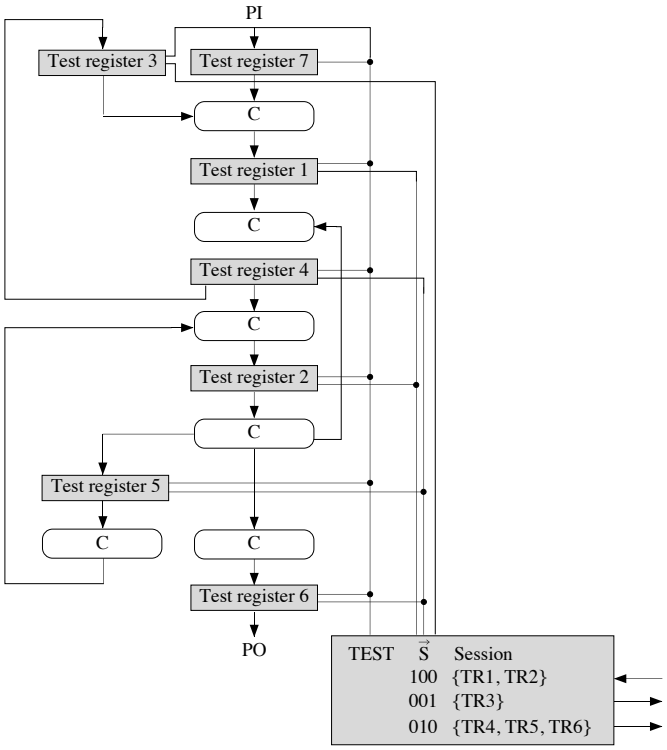


Figure 5: BIST control lines and their assignment

A test-per-clock scheme leads to short test times as a new pattern is generated in each clock cycle at least for a part of the circuit. A high speed test can be implemented at system frequency without any clock delays for shifting, and two pattern tests may be generated by appropriate test registers [8]. One drawback is that the test registers are larger than a scan path combined with a serial pattern generator, and integrating test registers into the data path has a stronger impact on system performance than integrating a scan path. In most cases, the

BIST control of a test-per-clock scheme is more complex than the BIST control of a test-per-scan scheme.

2.1 Pseudo-random pattern generators

Usually, test pattern generators and test response evaluators are implemented by *feedback shift registers* (Figure 6).

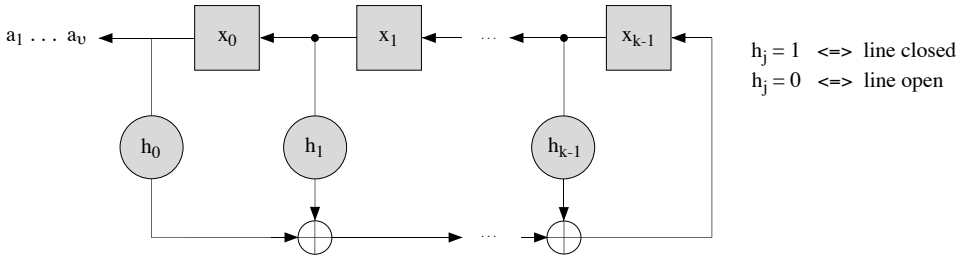


Figure 6: Standard linear feedback shift register (SLFSR)

The behavior of a *standard linear feedback register* (*SLFSR*) is completely determined by the feedback coefficients $h_0, \ldots, h_{k-1}$ which define a polynomial $h(X) := X^k + h_{k-1}X^{k-1} + \ldots + h_1 X + h_0 \in IF_2[X]$ called *characteristic* or *feedback polynomial*. From linear algebra we know that the state transition matrix

$$H \cdot \begin{pmatrix} x_0 \\ \vdots \\ x_{k-1} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \ldots & 0 & 1 \\ h_0 & h_1 & \ldots & & h_{k-1} \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ \vdots \\ x_{k-1} \end{pmatrix}$$

has the characteristic polynomial $\mathrm{X}_H(X) := \det(H + X \cdot ID) = h(X)$. The output sequence a_ν of the SLFSR must satisfies the recurrence equation

$$a_\nu = \sum_{j=0}^{k-1} a_{\nu-k+j} \cdot h_j \,.$$

The all-0-state cannot be part of such a random sequence which may have the maximum period of $2^k - 1$. For each $k \geq 1$ there is a sequence with this maximum period, the corresponding polynomials are called primitive and may be constructed algorithmically or found in tables [9,10].

If the feedback polynomial is primitive, the output sequence $(a_\nu)_{\nu \geq 0}$ has the some random properties [11] and is called pseudo-random. Pseudo-random patterns work well for testing in many cases but may also lead to reduced fault coverage due to linear dependencies. The

sequence $(a_\nu)_{\nu \geq 0}$ establishes a system of equations with variables $(x_0, \ldots, x_{k-1})$ from the initial state which may not be solvable. In the example of Figure 7 the fault $s0-Y$ requires $a_1 = a_3 = a_4 = 1$. This leads to the system of equations

$$\begin{aligned} x_1 &= 1 \\ x_0 + x_2 &= 1 \\ x_0 + x_2 + x_1 &= 1 \end{aligned}$$

for which no solution exists.

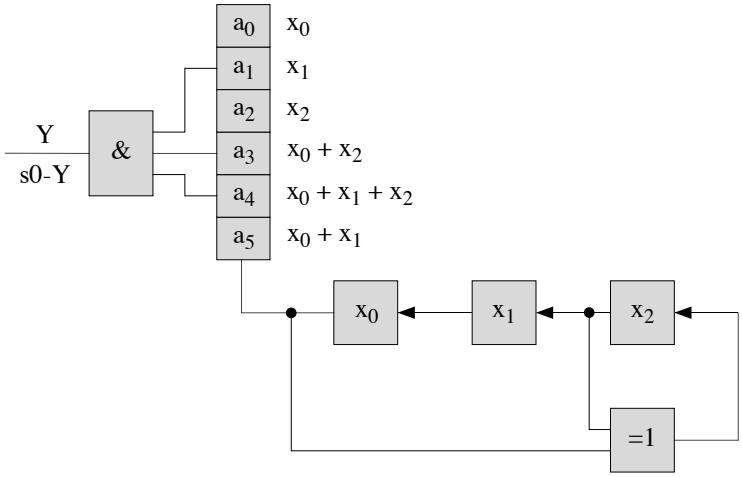


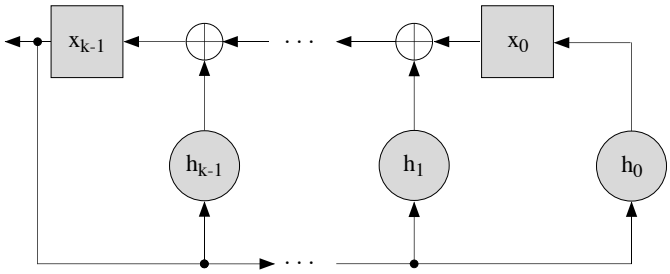
Figure 7: Testing an AND-gate

If M is a set of bit positions in the sequence $(a_\nu)_{\nu \geq 0}$ generated by an LFSR of length k, then the system of equations determined by M is linearly dependent with probability [12]:

$$P = 1 - \prod_{\mu=0}^{|M|-1} \frac{2^k - 2\mu}{2^k - \mu - 1}.$$

For example, selecting 20 bits from a 32-bit LFSR sequence leads to a probability of P=0.000244 that these 20 bits are dependent and cannot be set randomly.

LFSRs may also be implemented in a modular way as shown in Figure 8. The XOR-gates are distributed between the stages, the maximum delay is one XOR gate, and MLFSRs are faster than SLFSRs. Moreover, we have an increased perturbation of the internal state sequence which is useful for a test-per-clock scheme.



$$\begin{pmatrix} 0 & \dots & & 0 & h_0 \\ 1 & 0 & \dots & 0 & h_1 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \\ 0 & \dots & 0 & 1 & h_{k-1} \end{pmatrix}$$

Figure 8: Modular linear feedback register

Using the state transition matrices it can easily be proven that MLFSRs and SLFSRs have the same input/output behavior and are equivalent . Hence, all results concerning SLFSRs hold for MLFSRs, too. The decision on or against SLFSRs respectively MLFSRs for a test-per-scan scheme has to consider the high speed of an MLFSR in comparison with the more regular design style of the SLFSR. For a test-per-clock scheme the MLFSR has the additional advantage of the higher perturbation of the patterns.

2.2 Test response evaluation

As pseudo-random test lengths are rather long, the MUT responses cannot be compared with responses stored on chip but must be compressed into a single word by the TRE. In consequence, some information will be lost so that certain faulty response sequences may not be detected. This is called aliasing or fault masking.

The use of LFSRs for response compression as shown in Figure 9 is called signature analysis. A bit stream E is serially fed into the LFSR, the output stream is not observed, and only the state of the LFSR, called signature, is evaluated after the test.

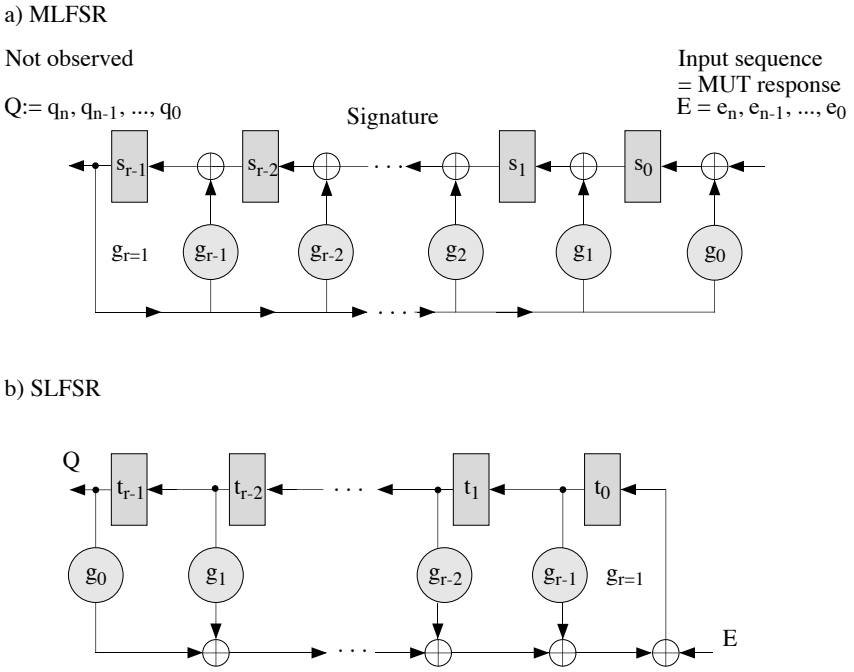


Figure 9: Signature analysis

The coefficients of the feedback function define the feedback polynomial $g(X) = g_r X^r + g_{r-1} X^{r-1} + \ldots + g_0$, and the input bit sequence defines both the input polynomial $e(X) = e_n X^n + e_{n-1} X^{n-1} + \ldots + e_0$ and the output polynomial $g(X) = q_n X^n + q_{n-1} X^{n-1} + \ldots + q_0$. The remainder polynomial $s(X) = s_{r-1} X^r + \ldots + s_0$ corresponds to the final state of the MLFSR.

Easily $\frac{e(X)}{g(X)} = q(X) + \frac{s(X)}{g(X)}$ is shown, hence signature analysis by an MLFSR is just a polynomial division, and the signature is the remainder. As MLFSRs and SLFSRs are equivalent, SLFSRs do polynomial division as well, but here the signature consists of the first coefficients t_i of the rational function $t(X) := \frac{s(X)}{g(X)}$.

Aliasing occurs if the faulty sequence and the correct sequence lead to the same signature. As each signature $s_{r-1}, \ldots, s_0$ corresponds to 2^{n-r} different response sequences, the correct signature corresponds to $2^{n-r} - 1$ different faulty response sequences. Under the simplifying assumption that all faulty sequences have the same probability, the aliasing probability is $\frac{2^{n-r} - 1}{2^n - 1} \approx 2^{-r}$. The formula is also true under more general conditions if the feedback polynomial is primitive and the errors occur randomly [13]. The analysis also holds for parallel signature analysis where multiple input sequences are fed into the LFSR [14].

A different compaction technique is *ones counting*, where the fault-free characteristic is the number of ones in the output stream, and *transition counting* means counting the number of 0-

1 and 1-0 transitions in the bit stream. Depending on the application, either ones counting, transition counting or signature analysis is the best solution. In most cases, signature analysis should be preferred, a comparison of the aliasing probabilities of these methods is found in [15].

The classic methods for pattern generation, response evaluation and BIST control do not apply to hard cores without scan path or without test registers. Even a hard core with scan design cannot be tested this way if it contains random pattern resistant faults whereas for soft cores and user defined logic there are modifications of the classic BIST solutions to be discussed next.

3 INTRODUCING BIST INTO MERGEABLE CORES AND USER DEFINED LOGIC

Some part of the user defined logic may be provided in the form a gate level netlist. Test methods which apply here may also be used for mergeable cores. Another part of the user defined logic is usually devoted to the application specific control logic to be synthesized from state transition tables or other behavioral descriptions. In both cases different BIST methods have to be applied.

3.1 Introducing BIST into structural netlists

LFSR-based BIST fails if certain faults have a very low detection probability, and the structural description of a soft core or a user defined logic at gate level or RT level can be modified to increase random pattern testability.

Random Pattern Testability: The cost of pseudo-random pattern testing is the number of patterns for which the circuit has to respond correctly in order to be correct with sufficient probability. The necessary test length depends on the probabilities by which randomly generated patterns detect the faults. Since the determination of fault detection probabilities is a very complex problem, approximation methods are used.

In general, sequential circuits are not random testable and must be transformed to combinational ones by integrating a scan path. For example, a stuck-at fault at the most significant bit of a 6-bit counter with reset at $D = 0$ requires $2^5 = 32$ times the input $D = 1$ Such a sequence has a probability as low as 2^{-32} and could not be generated randomly. Even in combinational networks we have random pattern resistant faults.

For an n-input circuit the *1-controllability* of an internal node k is the probability $p(k) = \frac{\text{Number of patterns which set } k = 1}{2^n}$ to set $k = 1$ randomly, the *0-controllability* $1 - p(k)$ is the probability to set $k = 0$ randomly, and the fault detection probability of a fault f is the probability

$$p_f = \frac{|T(f)|}{2^n}$$

to apply a test pattern from the complete test set $T(f)$ randomly. The *observability* of k is the probability of detecting a wrong value at k and can be computed as $p_{s0-k} + p_{s1-k}$.

Let N be the number of random patterns, and p_f the detection probability of fault f. Then $(1-p_f)^N$ is the probability that none of the patterns detects f. The probability that f is detected at least once is called the *confidence* of test and computed by $C := 1-(1-p_f)^N$. For a given confidence C the inequation

$$N \geq \frac{\ln(1-C)}{\ln(1-p_f)} \approx \frac{\ln(1-C)}{-pf}$$

determines the required test length.

Let F be a set of faults, all of them have the detection probability p. The test length N, required for detecting all faults, is (see [16])

$N \geq \frac{\ln(1-C) - \ln(|F|)}{p}$, and the expected fault coverage [17] is estimated by

$$1 - \frac{1}{|F|}\sum_{f \in F}(1-p_f)^N .$$

Hence, the test length depends logarithmically on the circuit size and on $(1-C)$, but depends linearly on $\frac{1}{p_f}$ which is 2^{-n} in the worst case. Obviously, some circuits and even logic functions are inherently not random pattern testable and need modifications by test point insertion [18, 19]. The test set for such a kind of circuit can be reduced considerably if weighted random patterns (WRPT) are generated, which set a one to each input of the circuit with a specific optimal probability [16-17,19-25].

Circular BIST

The synthesis of a test-per-clock scheme is implemented in the easiest way by a *circular BIST* or a *circular self-test path* [26]. The scheme has only two modes, the system mode and the test mode, where the flip-flops form the LFSR with feedback polynomial $x^n + 1$. Two arbitrary flip-flops may be the scan-in and scan-out inputs (Figure 10). In the test mode, the system performs signature analysis and pattern generation concurrently, and only a single control line T is required for the basic cells of this scheme (Figure 11).

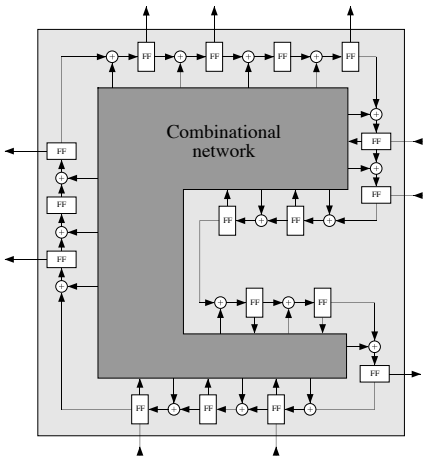


Figure 10: Circular BIST, circular self-test path

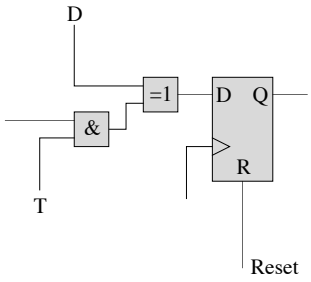


Figure 11: Basic cell for circular BIST

For this scheme, automation is as easy as scan design, the hardware overhead is very low, and no costs for BIST control are involved. Unfortunately, the scheme is not always applicable due to low fault coverage. Reasons may be a low random pattern testability of the combinational network., unreachable states, which are required as test patterns, or there may be only short cycles.

Sometimes the problems are alleviated by reordering the flip-flops or by introducing additional flip-flops into the circular path, or a more complex feedback polynomial than $x^n + 1$ is used. It may even be necessary to introduce an LFSR as a serial pattern generator (Figure 12), and if all these means do not help we have to look for a different BIST scheme.

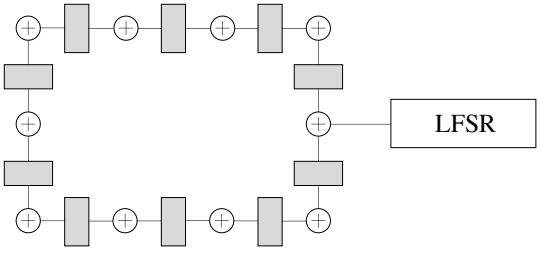


Figure 12: Adding an LFSR to a circular self-test path

A BIST scheme based on multi-functional test registers is considerably more complex since we have to cut self-loops of registers either by introducing a test register which is transparent in system mode or by using so-called CBILBOs. The CBILBOs are test registers which are able to perform signature analysis and pattern generation concurrently [27].

If a gate level netlist must be made self-testable, single flip-flops have to be clustered to test registers. The clustering should not introduce new cycles at RT-level as indicated in Figure 13.

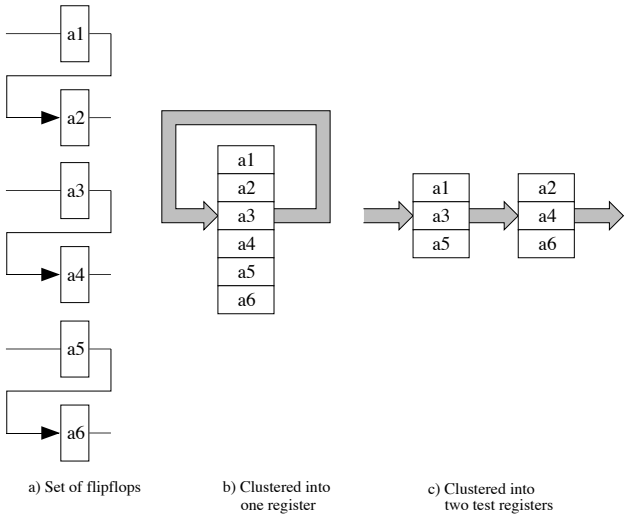


Figure 13: Clustering may destroy the BIST capability

If the six flip-flops are clustered into a single test register, a self loop is created and an expensive CBILBO or a transparent test register must be used (Figure 13 b), in Figure 13 c we get an easily testable RT-structure. Objectives of an efficient test register clustering are not only to introduce no new cycles but also a simple BIST scheduling and control [28].

3.2 Synthesis of self-testable control units

The classic techniques of FSM synthesis are described in [29], e. g., in a very comprehensive form. Innovative methods are found in [30], and in the following we only compile the most important approaches of current synthesis tools [31]. Figure 14 shows the structure of a synchronous control unit with the binary input variables $x_0, ..., x_{m-1}$, the output variables $z_0, ..., z_{r-1}$ and the state variables $y_0, ..., y_{n-1}$.

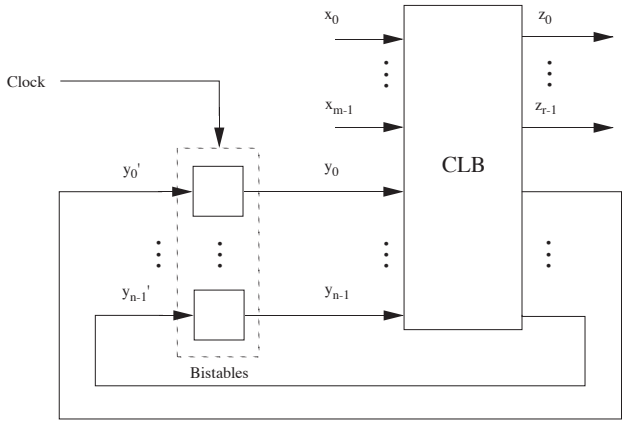


Figure 14: Structure of a control unit

The control unit of Figure 14 defines an automaton or a finite state machine $A = (S, I, O, \delta, \lambda)$ by $S := \{0,1\}^n$, $I := \{0,1\}^m$ and $O := \{0,1\}^k$. The combinational logic CLB implements the state transition function $\delta : I \times S \rightarrow S$ and the output function $\lambda : I \times S \rightarrow O$. A corresponding gate level structure is synthesized by the following steps [30-32]:

1) Behavioral transformation: The automaton A is mapped to an equivalent automaton A', or it is decomposed into an equivalent network of automata A'_i by means of algebraic structure theory.

2) State encoding: States, inputs and outputs of the automata A' or A'_i, rsp., are mapped to binary words. This gives us a description of the term $A' = (\{0,1\}^n, \{0,1\}^m, \{0,1\}^r, \delta', \lambda')$.

3) Logic synthesis: The uniquely defined Boolean functions λ' and δ' have to be implemented by combinational logic using logic synthesis tools.

Very often, step 1) is skipped and the automaton A is encoded and synthesized directly. The three steps are not independent, the behavioral transformations are performed in order to support state encoding and logic synthesis. State encoding has to consider the logic synthesis step and has to follow different heuristics if two-level or multi-level logic blocks are synthesized, e.g. The result is a structure as shown in Figure 15 or a system of interacting controllers of the same type.

Conventionally, the BIST is implemented in an extra „design-for-testability“ step after the synthesis of the circuit. The first task to make this structure self-testable is implementing the system register as test register. If the system register is also used as test pattern generator (TPG), an additional signature register has to be implemented providing a circuit structure as shown in Figure 15 b.

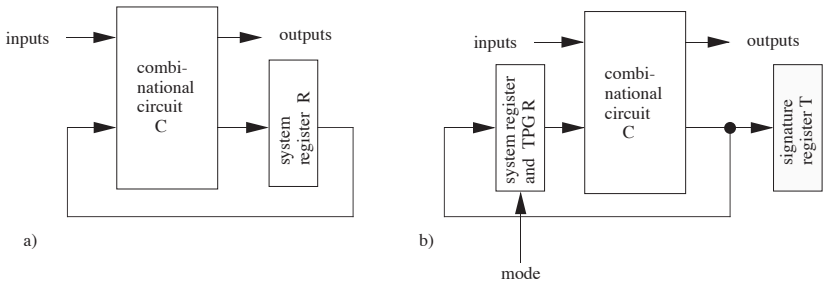


Figure 15: Controller structure obtained by conventional synthesis procedures (a) and required modifications for BIST (b).

The BIST implementation obtained by such a two-pass procedure has some serious drawbacks:

1) The number of flip-flops must be doubled, and extra logic is required for the multi-functional registers.
2) In system mode the signature register T must be transparent or bypassed using a demultiplexer. This prolongs the critical path and may slow down the system speed of the controller.
3) There are faults on the feedback lines from R to the inputs of C which are not detected as these lines are not completely exercised during self-test.

To overcome these disadvantages, a number of target structures and synthesis procedures for controllers has been developed taking into account the requirements of an efficient test-per-clock BIST. A so-called „one-pass“ synthesis method may follow different strategies:

- The BIST functionality is described at the behavioral level, too, and the corresponding BIST hardware is synthesized concurrently with the system hardware [33-35]. A simple example is the emulation of a scan path, where each state is reachable within n clock cycles, and the encoding of this state is shifted in at an additional input [36].

- The BIST hardware is also used for realizing parts of the system functionality [33].

- The synthesis supports and avoids certain target structures. For example self-loops are disadvantageous for a test-per-clock scheme and should be avoided [37].

LFSR-based self-testable controllers: Figure 15 b) shows a self-testable structure, where the direct feedback path from storage elements to storage elements via the combinational logic is broken by doubling the number of flip-flops and adding an additional self-test register for compacting the test responses. The state register itself is reconfigured as a pattern generator in self-test mode. Another possibility would be to include the MISR (multiple input signature register) in the feedback. These solutions are feasible if a small number of flip-flops has to be duplicated, but for highly sequential circuits they may result in significant hardware overheads.

The state registers of Figure 15 b) are not only D-flip-flops, but they have also the additional functionality of a pattern generator or a signature register. The following simple example shows how the ability of a linear feedback shift register (LFSR) to generate patterns can be utilized for implementing the system logic. Figure 16 a shows a state diagram of an FSM to be implemented. For test pattern generation the LFSR with the feedback polynomial $1 + x + x^2$ is used. Its autonomous state transitions are shown in Figure 16 b. It is easily seen that the LFSR function covers a part of the system function if the states are encoded properly. There is no need to implement these state transitions in the system logic if it is possible to switch the state register between D-flip-flop and LFSR mode in the synthesized circuit structure. To be useful, of course, the savings from not having to implement these state transitions have to be larger than the cost for the additional mode control signal.

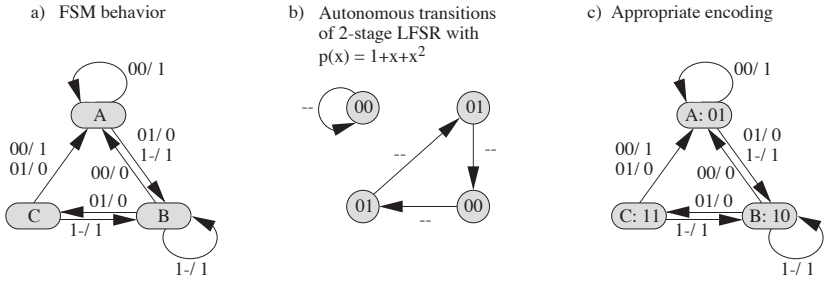


Figure 16: Example for utilizing the pattern generation capability of the state register

Pattern generators for self-testable designs cycle through a fixed sequence of states to stimulate the circuit. This property can also be used in system mode if the encodings of the present and the next state are consecutive elements in this cycle. Whenever the next state code is produced by the pattern generation register, which has to be implemented for testing purposes anyway, it is not necessary to generate it in the next state logic. Replacing the next state entries with "don't cares" for all such transitions, greatly increases the potential for logic optimization of the combinational logic. Figure 17 illustrates a possible realization of this idea [38]. An additional output signal „Mode" determines, whether the state machine flip-flops behave like ordinary D-flip-flops or function in pattern generation mode. In this mode the state register generates the next state on its own and the next state signals asserted by the combinational logic can be set to arbitrary values. Since in this structure pattern generation is integrated into system mode, we refer to it as *PAT*.

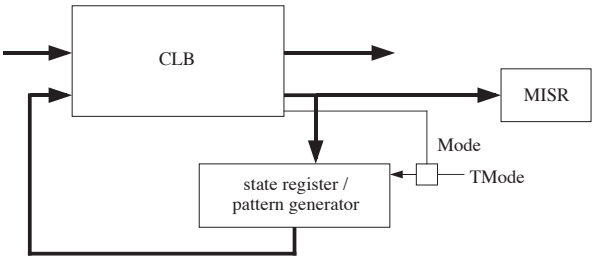


Figure 17: BIST structure with integrated pattern generator (PAT)

If a circular self-test path is a target of FSM synthesis, signatures are used as test patterns, and a parallel self-test can be carried out. For complete fault coverage, all states should be reachable in the BIST mode, too. In [39] it was already shown that a state transition graph may not be strongly connected in the test mode even if it is in the system mode, and some states may not be reachable any more. For improving fault coverage, they modify state encoding. Main drawbacks of this approach are the additional edges introduced in the state transition diagram and an encoding technique, which does not support logic synthesis, both may lead to significant hardware overhead.

As an alternative, Agrawal, Blanton and Damiani synthesize a parallel self-testable FSM without any MISR, and use the same type of register in system mode and in test mode [35]. During BIST, the machine runs through a state sequence autonomously, and the final state is evaluated. They extend the functional specification in such a way that each state corresponds to a successor input in the test mode, and a structure as shown in Figure 18 is generated.

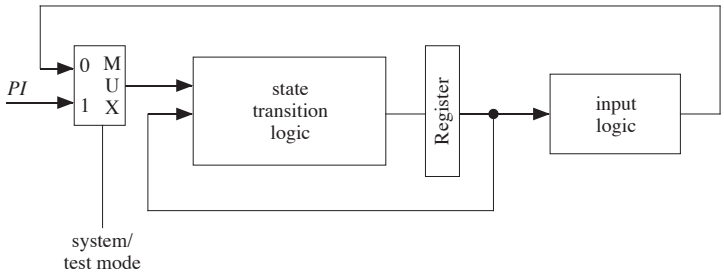


Figure 18: Self-testable control unit by [35]

The combinational blocks of the input logic and the state transition logic may be combined and synthesized together. If logic synthesis does not introduce redundancies, each state transition must be exercise for generating a complete test. For minimizing the test length and simplifying the implementation, the input logic block is chosen such that an Eulerian path is followed where each edge is traversed exactly once.

In contrast to the solutions presented above, the structure of Figure 19 does not contain a control signal for switching between test mode and system mode. Such a structure becomes possible if the system functionality is implemented by using the MISR in its signature analysis mode as a state register [33].

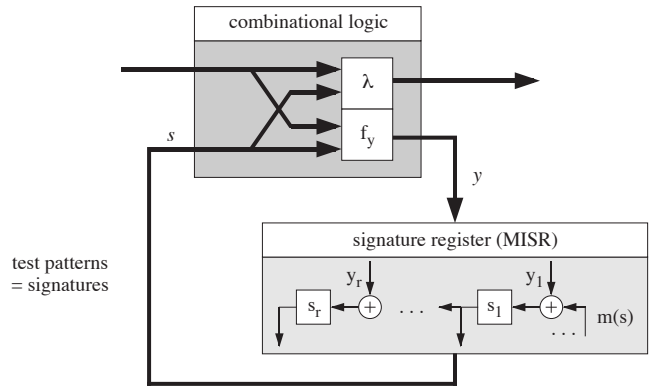


Figure 19: Parallel self-testable structure with integrated signature analysis (PST)

Let Hs be the next state of a MISR in autonomous mode, $m(s)$ the feedback function of the MISR, $\delta(i,s)$ the next state function of the system logic and $f_y(i,s)$ the excitation function of the state register. Because of the linearity of the operations involved, the necessary excitation variable y to produce a state transition from state s to state s^+ can easily be computed by $\delta(i, s) = f_y(i, s) + Hs$ and $y = f_y(i, s) = \delta(i, s) = s^+ + Hs$.

This is similar to T-flip-flops, where we have $\delta = (i, s) = y \oplus s = f_y(i, s)$. By implementing a pertinent next state function $f_y(i, s)$ in the combinational logic, arbitrary circuits can be implemented with MISRs as state registers, which makes it unnecessary to provide a special system mode.

In many cases the circuit structure for a parallel self-test without disjoint system and test modes has advantages with respect to area and testability. The reduced area is mainly due to the elimination of the D-flip-flop mode. Besides signature analysis the only other mode needed is a scan mode to initialize the flip-flops and to shift out the resulting signature. Therefore the number of control signals and the area of the self-test register are decreased, and at the same time a higher testability compared to other parallel self-test structures is obtained.

As there is no difference between system and test mode, all dynamic faults occurring in system mode can be detected during self-test. The test length and the effort to produce effective test patterns for the primary inputs may, however, increase. Hence, in this case, it is especially useful to apply logic synthesis methods, which maximize random pattern testability [40]. In summation there is no single self-test structure that is preferable in all cases. If automatic synthesis procedures are available for all self-test structures, it is possible to try alternative designs and then decide about the actual implementation of the circuit.

4 BIST OF A HARD CORE

As hard cores cannot be modified for incorporating BIST, we have to modify the pattern generator for improving the efficiency of the test sequences. Changing the seeds [41] or computing optimal seeds [42-43] is helpful if test patterns are not evenly distributed in the state sequence of the LFSR. Better results are obtained by changing the feedback polynomial or using multiple polynomials as in this case linear dependencies can be reduced or even exploited [44].

If the MUT contains random pattern resistant faults, more sophisticated methods have to be used. Weighted random patterns may be applied to circuits, where uniform pseudo-random testing would lead to an insufficient fault coverage. Weighted patterns are generated on the chip by feeding n independent random sequences into an n-input Boolean function. If a function has k minterms, the output sequence has probability $\frac{k}{2^n}$, and a 3-input AND gate generates a sequence of probability $\frac{1}{8}$, e.g.

4.1 Deterministic BIST

Recently, deterministic and mixed mode BIST schemes have attracted some attention. They are fault model oriented and generate precomputed test sets on chips. They first generate pseudo-random tests and add deterministic patterns, embed deterministic patterns or change random patterns. So far, the best results for a parallel deterministic BIST scheme are obtained by modifying the patterns generated by an LFSR [45,46-47]. A mapping α transforms some random patterns into test patterns as shown in Figure 20.

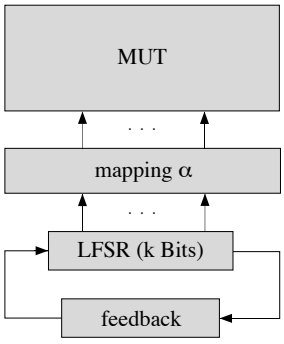


Figure 20: Modified LFSR sequences

The efficiency of the basic structure of Figure 20 is caused by the fact that not all bits of deterministic test patterns are specified. Usually, they contain a very large number of don't care bits to be used for optimizations [48]. In the sequel we estimate the number of bits of a

random pattern which have to be flipped in order to be compatible with an incompletely specified deterministic pattern.

Assume a scan path with n flip-flops and an LFSR generating the pseudo-random test set M of cardinality $m := |M|$. Let T be a deterministic pattern with s specified bits and $n-s$ unspecified bits. The probability that there is a pattern $T_d \in M$ which has a conflict with T at most at d bit positions, $d \le s$, is estimated by

$$P_d \approx \frac{m}{2^n} t_d, \text{ where } t_d = 2^{n-s} \sum_{i=0}^{d} \binom{s}{i}, \text{ while } m \cdot t_d < 2^n.$$

For $m \cdot t_d \ge 2^n$ the probability is nearly 1. The term t_d denotes the absolute number of patterns which have a conflict with T in no more than d bit positions. The above-mentioned formula can be transformed into

$$P_d \approx \frac{m}{2^s} \sum_{i=0}^{d} \binom{s}{i},$$

and the expectation value of the number d of bits to be flipped depends on m and s:

$$E(m,s) = \sum_{d=1}^{s} d \cdot \left(P_d - P_{d-1} \right)$$

Table 1 shows the expectation values for different random test sizes m and numbers of specified bits s.

m	s=10	s=20	s=30	s=40	s=50	s=60	s=70
1,000	0.02	2.78	6.09	9.54	13.32	17.17	21.11
10,000	0.00	1.79	4.66	7.83	11.39	15.03	18.74
100,000	0.00	0.90	3.53	6.50	9.65	13.19	16.64
1,000,000	0.00	0.05	2.54	5.21	8.29	11.52	14.89

Table 1: Expected number E(m,s) of bits to be flipped

For example, for a pattern with $s = 20$ specified bits we can expect to find one out of 10,000 random patterns, which has to be flipped at only two (≈ 1.79) positions. In general, the expected number of bits to be flipped in order to generate a precomputed test pattern is significantly smaller than the number of bits specified in that pattern. This effect can also be exploited for a test-per-scan scheme as shown in Figure 21 [49].

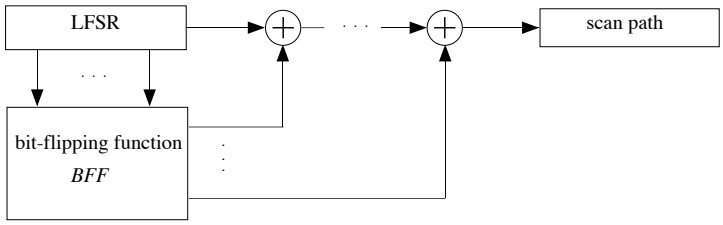


Figure 21: General form of bit-flipping BIST

A bit-flipping function must change the LFSR at a few bit positions, depending on the actual state of the LFSR. The bit-flipping function *BFF* has a very small off-set which corresponds to the useful random patterns, a very small on-set corresponding to bits to be flipped, and a very large don't care set. This results in a large potential for optimization which is exploited systematically in [49]. Even better results are obtained if the bit-flipping function receives its inputs not only by the LFSR states but also by the states of the BIST control unit BCU.

In [50-51] a mixed mode test-per-scan architecture has been presented which allows a very efficient encoding of the deterministic test vectors by LFSRs. It has been shown that a test pattern with s specified bits can be encoded into an s bit word with a very high probability of success. The s word is stored as a seed or a feedback function of a multiple-polynomial LFSR as introduced in [50] (see Figure 22). The LFSR can operate corresponding to a limited number of different feedback polynomials, and is used for both the generation of pseudo-random patterns and the decompression of encoded deterministic patterns.

A deterministic pattern is encoded as a polynomial identifier (abbreviated as ``id'' in Figure 22) and a seed for the respective polynomial. During test mode the pattern can be reproduced by establishing the feedback links corresponding to the polynomial identifier, loading the seed into the LFSR and performing m autonomous transitions of the LFSR. After the mth transition the scan chain contains the desired pattern which is applied to the CUT.

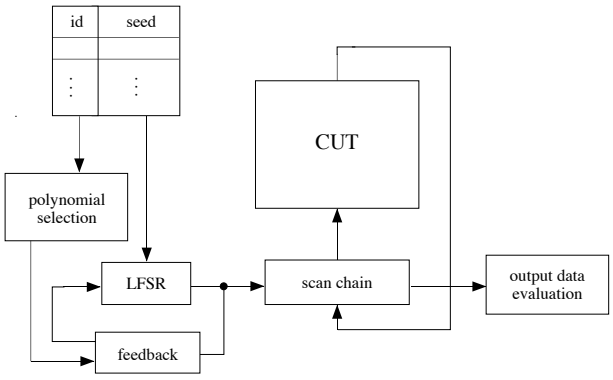


Figure 22: BIST scheme based on multiple polynomial LFSR

To calculate the encoding, systems of linear equations have to be solved. For a fixed feedback polynomial $h(X) = X^k + \sum_{j=0}^{k-1} h_j X^j$ of degree k the LFSR produces an output sequence $(a_i)_{i>0}$ satisfying the feedback equation $a_i = \sum_{j=0}^{k-1} a_{i-k+j} h_j$ for all $i \geq k$. The LFSR-sequence is compatible with a desired test pattern $t = (t_1, \ldots, t_m)$ if for all specified bits $a_i = t_i$ holds. Recursively applying the feedback equation provides a system of linear equations in the seed variables $a_0, \ldots, a_{k-1}$. If no solution can be found for the given polynomial the next available polynomial is tried, and in [50] it has been shown that already for 16 polynomials there is a very high probability of success that a deterministic pattern with s specified bits can be encoded into an s-bit seed. The identifier for the required feedback polynomial can be omitted if the seeds for specific polynomials are grouped together and a ``next-bit'' is used to indicate if the feedback polynomial has to be changed.

Hence, for encoding a deterministic test set $T = \{t_1, \ldots, t_N\}$ with a maximum number of specified bits $s_{\max} = \max\{s(t) \mid t \in T\}$ the seeds and the next bits require $(s_{\max} + 1) \cdot N$ bits of storage. If P polynomials are used, additional $s_{\max} \cdot P$ bits are needed for storing feedback taps, so that the overall storage requirements are $S(T) := (N + P)s_{\max} + N$ bits. Minimizing $S(T)$ requires minimizing both the maximal number of care-bits $s_{\max}$ and the number of patterns N. An ATPG method for minimizing the storage effort is presented in [51].

4.2 Exploiting the core functions for BIST

As processor kernels and programmable units are integrated into the system on chip, they can also be used for pattern generation and response evaluation. A way to program an embedded processor so that it generates a mixed mode test is found in [44]. The processor emulates an LFSR based pseudo-random test first, and after that it emulates the reseeding scheme described above.

Even simple accumulator based structures can work in an autonomous mode for generating patterns with some pseudo-random or pseudo-exhaustive properties (see Figure 23) or may compress test data like an LFSR during signature analysis.

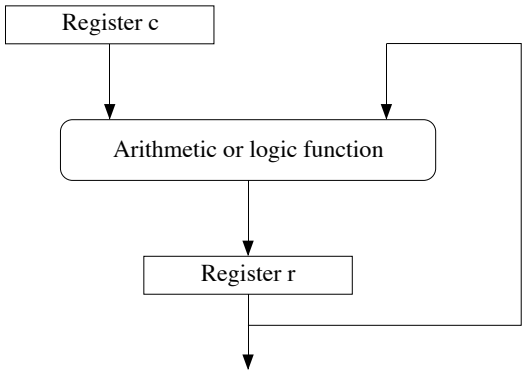


Figure 23: A typical accumulator structure used as test pattern generator. In each cycle the constant content of register *c* is added to register *r*. The content of register *r* is a test pattern

The advantages of this approach are twofold: as a specialized BIST circuitry is not needed, the hardware overhead is reduced to some modifications for implementing BIST control, and since BIST circuitry within the data path is completely avoided, this BIST method will not affect system performance.

The use of accumulator based structures for test response compaction leads to aliasing probabilities which have the same magnitude as the aliasing probabilities of the LFSR-based signature analysis [52].

Test pattern generation may be performed by using a variety of functional units in the accumulator based structure of Figure 23. Investigations are known about the test properties of patterns generated by simple adders [53], ones -and twos-complemented subtractors [54-55] and more complex multipliers and MAC circuits [52]. All of them may generate pseudo-exhaustive or pseudo-random patterns with a similar quality as LFSRs do and may reach a comparable fault coverage. Recently, a method for accumulator based deterministic BIST has been described in [56].

5 STANDARDIZATION

For several years building testable systems and boards has been supported by the IEEE 11149 standards which describe test means and interfaces of discrete chips on a board or an MCM. These standards do not apply well to core based systems since now the core user is responsible for manufacturing and testing the core. As the cores are delivered from multiple sources, different test views and paradigms have to be merged into a single test strategy. Current standardization efforts try to define how the internal design-for-test and BIST structures of a core should be made available, how the core periphery should be made accessible and how test and diagnosis of the complete systems should be supported. A working group is elaborating a proposal IEEE P 1500 which will be voted in 1999.

References

[1] E.B. Eichelberger and E. Lindbloom, Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test, IBM Journal of Research and Development **27 (**3) (1983).

[2] P.H. Bardell and W.H. McAnney, Self-testing of multichip logic modules, Proc. IEEE International Test Conference (1982) pp. 200-204.

[3] B. Koenemann et. al., Built-In Logic Block Observation Techniques, Proc. Test Conference, Cherry Hill, New Jersey (1979).

[4] G.L. Craig, C.R. Kime and K.K. Saluja, Test Scheduling and Control for VLSI Built-In Self-Test, IEEE Transactions on Computers, September (1988) 1099-1109.

[5] A.P. Stroele and H.-J. Wunderlich, Signature Analysis and Test Scheduling for Self-Testable Circuits, Proc. International Symposium on Fault-Tolerant Computing, Montreal (1991) pp. 96-103.

[6] O.F. Haberl, H.-J. Wunderlich, The Synthesis of Self-Test Control Logic, Proc. COMPEURO (1989) pp. 5.134-5.136

[7] Y. Zorian, A Distributed BIST Control Scheme for Complex VLSI Devices, Proc. VLSI Test Symposium (1993) pp. 4-9

[8] P. Girard, C. Landrault, V. Moréda and S. Pravossoudovitch, An Optimized BIST Test Pattern Generator for Delay Testing, Proc. 15th VLSI Test Symposium, April (1997) pp. 94-99.

[9] W. W. Peterson and E.J., Jr. Weldon, Error-Correcting Codes MIT-Press, Cambridge, Massachusetts, London (1972).

[10] R. Lidl and H. Niederreiter, Introduction to finite fields and their applications, Cambridge, Cambridge University Press (1986).

[11] S.W. Golomb, Shift Register Sequences, Aegan Park Press, Laguna Hills (1982).

[12] C.L. Chen, Linear Dependencies in Linear Feedback Shift Registers, IEEE Transactions on Computers C-35 (12) (1986) 1086-1088.

[13] T.W. Williams, W. Daehn, W. Gruetzner and C.W. Starke, Comparison of Aliasing Errors for Primitive and Non-Primitive Polynomials, Proc. IEEE International Test Conference, Philadelphia, September (1986) pp. 282-288.

[14] M. Damiani et. al., Aliasing in Signature Analysis Testing with Multiple-Input Shift-Registers, Proc. 1st European Test Conference, Paris (1989) pp. 346-353.

[15] M. Abramovici, M.A. Breuer and A.D. Friedman, Digital Systems Testing and Testable Design, IEEE PRESS, revised printing (1990).

[16] H.-J. Wunderlich, PROTEST: A Tool for Probabilistic Testability Analysis, Proc. 22nd ACM/IEEE Design Automation Conference, Las Vegas (1985) pp. 204-211.

[17] R. Lisanke, F. Brglez A.J. DeGeus, D. Gregory, Testability Driven Random Test-Pattern Generation, IEEE Transactions on CAD, CAD-6 (6) Nov. (1987) 660-669.

[18] B. Krishnamurthy, Hierarchical Test Generation: Can AI Help?, Proc. IEEE International Test Conference, Washington D.C. (1987) 694-700.

[19] M. Bershteyn, Calculation of Multiple Sets of Weights for Weighted Random Testing; Proc. IEEE International Test Conference, Washington D.C. (1993) pp. 1031-1040.

[20] R. Krieger, B. Becker and R. Sinkovic, A BDD-based Algorithm for Computation of Exact Fault Detection Probabilities, Proc. 23rd International Symposium on Fault-Tolerant Computing (1993) pp. 186-195.

[21] J.A. Waicukauski, E. Lindbloom, E.B. Eichelberger and O.P. Forlenza, A Method for Generating Weighted Random Test Patterns, IBM Journal of Research and Development, 33 (2) March (1989) 149-161.
[22] R. Kapur, S. Patil, T.J. Snethen and T.W. Williams, Design of an Efficient Weighted Random Pattern Generation System, Proc. IEEE International Test Conference (1994) pp. 491-500.
[23] F. Muradali, V.K. Agarwal and B. Nadeau-Dostie, A New Procedure for Weighted Random Built-In Self-Test, Proc. IEEE International Test Conference (1990) pp. 660-669.
[24] S. Pateras and J. Rajski, Cube-Contained Random Patterns and their Application to the Complete Testing of Synthesized Multi-level Circuits, Proc. IEEE International Test Conference (1991) pp. 473-482.
[25] I. Pomeranz and S.M. Reddy, 3-Weight Pseudo-Random Test Generation Based on a Deterministic Test Set for Combinational and Sequential Circuits; IEEE Transactions on CAD 12 (7) (1993) 1050-1058.
[26] A. Krasniewski and S. Pilarski, Circular Self-Test Path: A Low Cost BIST Technique of VLSI Circuits, IEEE Transactions on Computer-Aided Design, Jan. (1989) 46-55.
[27] L.T. Wang and E.J. McCluskey, Concurrent Built-in Logic Block Observer (CBILBO), Proc. International Symposium on Circuits and Systems (1986) pp. 1054-1057.
[28] A. Stroele and H.-J. Wunderlich, Configuring Flip-Flops to BIST registers, Proc. IEEE International Test Conference, Washington D.C. (1994) pp. 939-948.
[29] Z. Kohavi, Switching and Finite Automata Theory, McGraw-Hill Book Company, New York, 2nd Edition (1978).
[30] P. Ashar, S. Devadas and A.R. Newton, Sequential Logic Synthesis, Kluwer Academic Publishers, Boston (1992).
[31] E.M. Sentovich et. al., SIS: A System for Sequential Circuit Synthesis, UCB Electronics Research Laboratory, No. UCB/ERL M92/40 Memorandum (1992).
[32] T. Sasao (Ed.), Logic Synthesis and Optimization, Kluwer Academic Publishers, Boston (1993).
[33] B. Eschermann and H.-J. Wunderlich, Parallel Self-Test and the Synthesis of Control Units, Proc. 2nd European Test Conference, Munich (1991) pp. 73-82.
[34] V.D. Agrawal and K.-T. Cheng, State Assignment for Testable Design, International Journal of Computer Aided Design 3 Mar. (1991).
[35] V.D. Agrawal, R.D. (Shawn) Blanton and M. Damiani, Synthesis of Self-Testing Finite State Machines from High-Level Specification, Proc. IEEE International Test Conference, Washington D.C. (1996) pp. 757-766.
[36] S.M. Reddy and D.S. Ha, A New Approach to the Design of Testable PLAs, IEEE Transactions on Computers C-36 201-211.
[37] S. Hellebrand and H.-J. Wunderlich, Synthesis of Self-Testable Controllers, Proc. EDAC/ETC/EuroAsic, Paris (1994) pp. 580-585.
[38] B. Eschermann and H.-J. Wunderlich, Optimized Synthesis Techniques for Testable Sequential Circuits, IEEE Transactions on Computer-Aided Design 11 (3) (1992) pp. 301-313.
[39] C.C. Chuang and A.K. Gupta, The Analysis of Parallel BIST by the Combined Markov Chain (CMC) Model, Proc. IEEE International Test Conference, Washington D.C. (1989) pp. 337-343.

[40] N.A. Touba and E.J. McCluskey, Automated Synthesis of Random Pattern Testable Circuits, Proc. IEEE International Test Conference (1994) pp. 174-183.
[41] J. Savir and W.H. McAnney, A Multiple Seed Linear Feedback Shift Register, IEEE Transactions on Computer, Feb. (1992) pp. 250-252.
[42] M. Lempel, S.K. Gupta, M.A. Breuer, M.A.: Test Embedding with Discrete Logarithms; IEEE Transactions on CAD o. Integrated Circuits a. Systems, 14 (5) May (1995) pp. 554-566.
[43] S.K. Mukund, E.J. McCluskey and T.R.N. Rao, An Apparatus for Pseudo-Deterministic Testing, Proc. 13th VLSI Test Symposium, Princeton, NJ (1995) pp. 125-131.
[44] S. Hellebrand, H.-J. Wunderlich and A. Hertwig, Mixed-Mode BIST Using Embedded Processors, Proc. IEEE International Test Conference, Washington D.C. (1996) pp. 195-204.
[45] N.A. Touba, E.J. McCluskey, Altering a pseudo-random bit sequence for scan-based BIST, Proc. IEEE International Test Conference, Washington D.C. (1996) pp. 167-175.
[46] S.B. Akers and W. Jansz, Test Set Embedding in Built-in Self-Test Environment, Proc. IEEE International Test Conference, Washington D.C. (1989) pp. 257-263.
[47] M. Chatterjee, D.K. Pradhan, A Novel Pattern Generator for Near-Perfect Fault-Coverage, Proc. 13th VLSI Test Symposium, Princeton, NJ (1995) pp. 417-425.
[48] B. Koenemann, LFSR-Coded Test Patterns for Scan Designs, Proc. European Test Conference, Munich (1991) pp. 237-242.
[49] H.-J. Wunderlich and G. Kiefer, Bit-Flipping BIST, Proc. IEEE/ACM International Conference on CAD-96, San Jose, CA, Nov. (1996) pp. 337-343.
[50] S. Hellebrand, S. Tarnick, J. Rajski and B. Courtois, Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers, Proc. IEEE International Test Conference, Baltimore, MD, Sept. (1992) pp. 120-129.
[51] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman and B. Courtois: Built-in Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers, IEEE Transactions on Computers 44 (2) Feb. (1995) 223-233.
[52] J. Rajski and J. Tyszer, Test Responses Compaction in Accumulators with Rotate Carry Adders, IEEE Transactions on CAD of Integrated Circuits and Systems 12 (4) Apr. (1993) 531-539.
[53] S. Gupta, J. Rajski and J. Tyszer, Arithmetic Adaptive Generators of Pseudo-Exhaustive Test Patterns, IEEE Transactions on Computers, 8 (45) Aug. (1996) 939-949.
[54] A.P. Stroele, Arithmetic Pattern Generators for Built-In Self-Test, Proc. International Conference on Computer-Aided Design (1996) pp. 131-134.
[55] A.P. Stroele, BIST Pattern Generators using Addition and Subtraction Operations, Journal of Electronic Testing: Theory and Applications, JETTA, 11 (1) Aug. (1997) 68-80.
[56] H.-J. Wunderlich, R. Dorsch, Accumulator Based Deterministic BIST, Proc. IEEE International Test Conference, Washington D.C. Oct. (1998).

Short biography of the author

Hans-Joachim Wunderlich received the Dr. rer. nat. (Ph. D.) degree in computer science from the University of Karlsruhe in 1986. There he was the head of a research group on automation of circuit design and test from 1986 to 1991. From 1991 to 1996 he was a full professor for computer science at the University of Siegen. Since October 1996 has been the head of the Division for Computer Architecture at the University of Stuttgart.

He has been a member of the program committee at numerous conferences and a reviewer of research proposals submitted to NSF and NATO. Within the European projects EUROCHIP and EUROPRACTICE he has been a lecturer for courses on VLSI design and test. Dr. Wunderlich is author and co-author of three books and over 70 papers in the field of test, synthesis, and fault tolerance of digital systems.