International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 07 | July - 2022 Impact Factor: 7.185 ISSN: 2582-3930

Storage-Based Built-In Self-Test for Gate-Exhaustive Faults and Delay Faults

Mrs. VASUNDHARA.P, Mr. MURUGANANDAN
Nehru College of Engineering, Pampady,Thrissur,Kerala

Abstract

Built-in self-test (BIST) approaches are suitable for in-field testing since they do not require a tester for storage and application of test data. They also reduce the security vulnerabilities associated with loading and unloading of external test data into scan chains. As technologies evolve, in-field testing needs to address more complex defect and aging mechanisms that require specific deterministic tests. This can be addressed by BIST approaches that store test data on-chip, and use the data for on-chip generation of both random and deterministic tests. In this case, there is a tradeoff between the amount of stored test data and the comprehensiveness of the test set that can be applied. The proposed work explores this tradeoff in a specific context that has the following main features. (1) The initial stored test data is based on a stuck-at and path delay test set. (2) The target faults are single-cycle gate-exhaustive faults. (3) The stored test data is enhanced gradually by test data based on a gate-exhaustive test set to increase the coverage of gate-exhaustive faults.

1.INTRODUCTION

Built-in self-test (BIST) approaches are suitable for in-field testing since they do not require a tester for storage and application of test data [1]-[10]. They also reduce the security vulnerabilities associated with loading and unloading of external test data into scan chains [3]. As technologies evolve, in-field testing needs to address more complex defect and aging mechanisms that require specific tests [11]- [14]. This can be addressed by BIST approaches that store test data on-chip, and use the data for on-chip generation of both random and deterministic tests. Different from hybrid approaches that combine test data compression with on-chip test generation [15]-[18], a BIST approach stores all the test data on-chip. This is the type of approach considered in this paper. In this approach, there is a tradeoff between the amounts of stored test data and the comprehensiveness of the test set that can be applied. The paper explores this tradeoff in the following context. The circuit under consideration has n scan chains of length l. One component of the stored test data is a set Si of scan vectors. Using only scan vectors from Si, an on-chip test generation logic applies a fixed number of tests referred to as random, where scan vectors are selected randomly from Si, and a small number of tests referred to as deterministic that consist of specific scan vectors from Si. The set of deterministic tests is denoted by Ti,dtrm, and it requires additional storage of test data (indices of scan vectors from Si that form the tests in Ti,dtrm).

The test set produced on-chip is denoted by Ti, and it changes with Si and Ti,dtrm. The on-chip test generation logic is the same for every Si and Ti,dtrm, and every circuit. Only its parameters (e.g., memory and multiplexer sizes) are circuit-dependent. A deterministic test set Tsa for single stuck-at faults is used as an initial source for test data. The test data is a set S0 of scan vectors based on Tsa, and a set T0,dtrm of deterministic tests to complement the random tests based on S0. With S0 and T0,dtrm, and the on chip test generation logic, T0 detects all the detectable single stuckat faults. The set of single stuck-at faults is denoted by Fsa. The effectiveness of the random tests is enhanced by using stored scan vectors from Tsa.

In addition to stuck-at faults, the set of target faults includes single-cycle (static) gate-exhaustive faults [19]. These are used for representing the need to cover a large number of defects with specific behaviors that are more difficult to detect than stuck-at faults. Gate exhaustive faults are suitable for this purpose since their number is large, and they require more values to be assigned by a test. The set of detectable gate-exhaustive faults is denoted by Fgexh, and a deterministic test set for gate-exhaustive faults is denoted by Tgexh. It is also possible to consider a test set for transition faults as an initial source for test data, and two-cycle (dynamic) gate-exhaustive faults as additional target faults.

Although over the next years, the primary objective of manufacturing test will remain essentially the same to ensure reliable and high quality semiconductor products conditions and consequently also test solutions may undergo a significant evolution. The semiconductor technology, design characteristics, and the design process are among the key factors that will impact this evolution. With new types of defects that one will have to consider to provide the desired test quality for the next technology nodes such as 3-D, it is appropriate to pose the question of what matching

design-for-test (DFT) methods will need to be deployed. Test compression, introduced a decade ago, has quickly become the main stream DFT methodology. However, it is unclear whether test compression will be capable of coping with the rapid rate of technological changes over the next decade. Interestingly, logic built-in self-test (LBIST), originally developed for board, system, and in-field test, is now gaining acceptance for production test as it provides very robust DFT and is used increasingly often with test compression. This hybrid approach seems to be the next logical evolutionary step in DFT. It has potential for improved test quality; it may augment the abilities to run at-speed power aware tests, and it can reduce the cost of manufacturing test while preserving all LBIST and scan compression advantages.

Serial or scan-based built-in self-test (BIST) offers an excellent solution for the challenges of today's integrated circuit testing. The built-in capabilities of test pattern generation and test response evaluation allow an efficient test even for externally inaccessible components in complex systems-on-a-chip (SOCs). The classical architecture with an LFSR feeding pseudo-random patterns into the scan path is easy to implement and minimizes both hardware overhead and the impact on the system performance.

2.BASIC CONCEPT OF BIST

Built-in-self-test (BIST) is a technique in which additional circuitry is added to core under test (CUT) in order to make it able to test itself with minimum external help. General structure of a self-testable circuit composed of test pattern generator (TPG) , a test response evaluator (TRE) and a BIST control unit (BCU) .This technique is especially preferable when it is difficult to access the CUT externally. It also helps to protect intellectual property (IP) and to reduce cost of the external test equipment (ATE) by minimizing the amount of test data that has to be stored off-chip. it implementation can result in an improvement in the test quality due to its better support for at-speed testing, which is essential for detecting delay faults. BIST supports in-field and on-line testing, which helps to reduce the cost of system maintenance. It also offers the opportunity to improve reliability by means of burn-in testing. BIST approaches can be divided into test-per-scan and test-per-clock schemes.

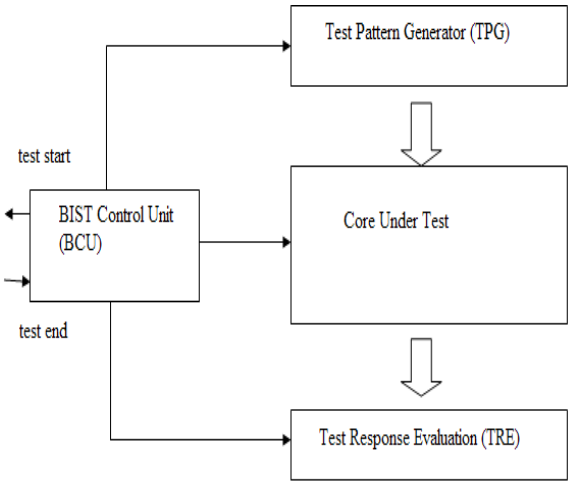


Fig. 1.1 BIST

2.1.TEST-PER-SCAN SCHEMES

Test-per-scan BIST schemes require scan-based design. In the case of sequential circuits, this means that all the storage cells can be configured as one or several scan paths (chains), which are used as serial shift registers in test mode .In this way, each storage device of the CUT becomes easily controllable and observable. The test stimuli/responses are shifted into/out of the scan paths. Scan-based design helps to reduce the problem of testing sequential circuits to the simpler problem of testing combinational circuits. The shift counter controls the bit stream which is generated and shifted into the scan path by a TPG.

The pattern counter controls the length of the test sequence. A system clock cycle (also called capture or functional clock cycle) is applied to load the CUT response to the current test pattern into the scan path. During the so-called shift mode (also called scan or test mode) a new test pattern is shifted into the scan path, while the CUT response to the previous pattern is shifted out and compressed by a TRE.

2.2.TEST-PER-CLOCK SCHEMES

In a test-per-clock scheme a test pattern is applied to the CUT every clock cycle. This scheme is best suited for register-based design. This kind of scheme employs a specific BIST architecture using the built-in logic block observer (BILBO),which is a more sophisticated register that can function as a normal state register, scan register, PRPG or MISR All functionality of the BILBO depends on the mode input signals .

2.3.TEST PATTERN GENERATION

Test pattern generation for both test-per-scan and test-per-clock BIST schemes can be classified into the following groups: pseudo-random, weighted, exhaustive, pseudo exhaustive, deterministic and mixed-mode schemes.

2.4. PSEUDO-RANDOM PATTERN GENERATION

Pseudo-random pattern testing is an attractive approach for BIST. Possible choices for pseudo-random pattern generators (PRPGs) are one-dimensional linear hybrid cellular automata (LHCAs), linear feedback shift registers (LFSRs) or different accumulator based structures As processor kernels or programmable units are integrated into SOCs, they can also be used for pattern generation . An LHCA is a collection of memory cells x1, x2,…, xj-1, xj, xj+1, …connected in such a way that each cell is restricted to local neighborhood interactions. The next state of each cell is determined based on the states of the cells with each the considered cell interacts. For example, if cell j can communicate only with the neighbor cells, j-1 and j+1, one of the following two rules can be employed: xj(t+1) =xj-1(t) Å xj+1(t) or xj(t+1) = xj-1(t) Å xj(t) Å xj+1(t), where xj(t) represents the state of cell j at time t. LFSR is a Moore finite state machine that consists of interconnected memory elements, also referred to as stages or cells, and linear logic elements such as exclusive-OR (XOR) or exclusive-NOR (XNOR) gates.

2.5.WEIGHTED-RANDOM PATTERN TESTING

Although LFSRs, LHCAs or other linear TPGs can generate a large set of pseudorandom test patterns with very simple hardware, this seldom provides sufficient fault coverage for a CUT. A way to address this problem is to use weighted-random pattern testing techniques. The TPG used in weighted-random pattern testing is composed of an LFSR and additional combinational logic to modify the probability of ones and zeros in the output sequence. This weighting circuitry is used to bias the pseudo-random patterns towards those that detect random pattern resistant faults, such that the fault coverage is increased and the test length can be reduced. Several techniques have been proposed for computing weight sets. Multiple weight sets are required to achieve sufficient fault coverage. For this reason, the weight sets have to be stored on-chip and additional control logic is needed to switch between them during the test time. This increases the BIST overhead a lot.

2.6. EXHAUSTIVE AND PSEUDO-EXHAUSTIVE TESTING

Exhaustive testing applies all possible 2n test patterns to an n-input combinational circuit, so that a high quality test can be obtained and no particular fault model is used. The test pattern generator can be a binary counter or an LFSR with a primitive feedback polynomial, in which the all-zero pattern may be generated by a reset signal. As the number of test patterns increases exponentially with the number of the circuit inputs, this approach is usually not feasible for circuits with a large number of inputs (n>30). Pseudo-exhaustive testing relies on the partition of the CUT into output cones which are tested exhaustively .As compared to exhaustive testing, far fewer test patterns are required. Evertheless, the feasibility of pseudo-exhaustive testing depends on the size of the largest output cone.

3. DETERMINISTIC TESTING

Deterministic testing applies a pre-computed set of test cubes (test patterns with unspecified bits) to the CUT. Thus, any coverage of the testable faults can be achieved. The patterns may be stored on-chip, e.g. using a ROM, or off-chip in which case they have to be loaded from an ATE. In both approaches the data volume to be stored tends to be extremely large. In the case of the ATE-based approach this may also have a strong impact on the required bandwidth. In order to reduce the storage and bandwidth requirements, special algorithms for generating compact test. Similar approaches can also be used with (ROM based) BIST schemes to reduce the storage requirements. Such methods are often called store and generate. An intensively investigated store and generate technique uses LFSR-reseeding. It is based on storing pre-computed LFSR seeds that can be used to generate deterministic test cubes]. Reseeding-based encoding provides a higher compression ratio than any other entropy-based compression method [Tou04]. As seeds are smaller than the test patterns themselves, they require less ROM storage. A small LFSR with a single feedback polynomial may not always have a seed that will generate all the required deterministic test cubes. Multiple-polynomial LFSR schemes can fix this problem. The LFSR can operate corresponding to a limited number of different feedback polynomials and produce all the deterministic cubes. Both polynomial and seed identifiers need to be stored. A different class of reseeding techniques is based on special counters that generate a deterministic set of test cubes. Twisted-ring

counter and folding counter are approaches which embed deterministic cubes into counter sequences.

They can efficiently reduce test data storage with full fault coverage, but the approaches are not compatible with standard scan design. More efficient compression and decompression methods are those in which a small amount of external test data is continuously fed into the chip. As long as these methods are based on the use of an external ATE and not on an internal memory, they are no longer BIST methods and lose some specific benefits of BIST like in-field and on-line testing.

3.1.MIXED-MODE TESTING

Mixed-mode approaches can achieve more efficient test data compression and hardware implementation than pure deterministic test schemes. Mixed-mode testing combines pseudo-random testing with various deterministic testing schemes so that the test storage requirements can be significantly reduced and high levels of fault coverage can be obtained within a reasonable test application time.

Usually in mixed-mode approaches, the pseudo-random patterns produced by LFSRs are used to test easy-to-detect faults. To increase the number of detected faults, test points can be inserted into the CUT. Deterministic test patterns can be generated by an automatic test pattern generator (ATPG) and stored in a ROM. In other mixed-mode approaches, often called test set embedding schemes, deterministic test patterns are embedded in pseudo-random sequences with the help of some additional combinational logic.

In the bit-flipping approach, the output sequence of an LFSR is inverted at a few bit positions in order to increase fault coverage, while the bit-fixing approach applies constant values. The so-called Star Test approach introduced in uses deterministic test patterns which are surrounded at a limited Hamming distance by clusters of child patterns. Based on the use of parent patterns, the Star Test approach can be considered a deterministic method. Due to the way in which the clusters of child patterns are produced, this scheme can also be classified as a generalized weighted-random pattern testing.

3.2. TEST RESPONSE EVALUATION

Besides test pattern generation, BIST architectures should also be able to compress/evaluate test responses. As the number of test patterns applied to the CUT is usually very large, it is infeasible to store all the expected values on-chip and compare them with the response values. It is much cheaper in terms of storage requirement and compacting circuitry to compress the test responses to short sequences, called signatures, which are delivered for analysis at the end of the test session. A signature is obtained as the final state of a finite state machine whose inputs are fed with test responses. This type of compression which addresses the length of the test response sequence is also known as time compression. Examples of time compressors are accumulator, LFSR- and counter-based compactors.

4. METHODOLOGY

4.1. DELAY FAULT MODELS

Defects that cause the faulty timing behavior of a circuit are modeled by delay faults. Two types of delay fault models are commonly used:

- the transition fault model
- the path delay fault model

4.1.1 TRANSITION FAULT MODEL

The transition fault model captures delay defects that cause a slow-to-rise transition or a slow-to-fall transition at a specific line in the circuit.A slow-to-rise transition fault at line c in a 3-input circuit. Input b and d have constant values. The value of input a changes from 0 to 1 at time point t1, i.e. a rising transition occurs at a, and the transition propagates through the circuit. If the circuit is fault free, the value of output e is 1 at the required time point t2, where t2-t1 is the clock period. However, due to the slow-to-rise transition fault at line c, the value of output e remains 0 at t2.

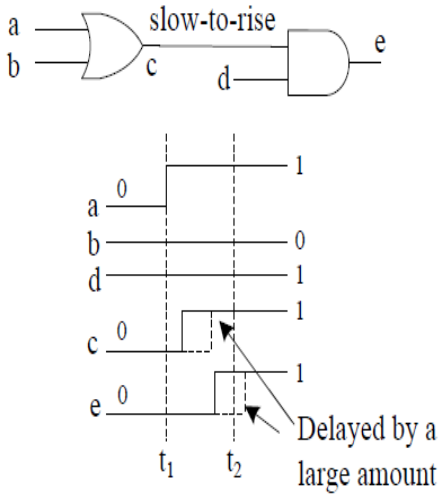


Fig.3.1.transition fault model

4.1.2 PATH DELAY FAULT MODEL

The path-delay fault is an important fault model used in delay testing. The delay defect in the circuit is assumed to. cause the cumulative delay of a combinational path to exceed some specified duration. Delay faults cause errors in the functioning of a circuit based on its timing. ... These tests can also be used to determine the proper clock frequency that the circuit could be run at and still have the circuit function correctly. The faults caused by the rise and fall times are called transition delay faults. Different from the transition fault model which only captures single large delay at a specific line, the path delay fault model captures small extra delays whose cumulative effect along a path from inputs to outputs may result in faulty behavior of the circuit.A path delay fault associated with path a-c-e-g and a rising transition at its source a in a 4-input circuit. Input b, d and f have constant values. A rising transition occurs at input a at time point t1, and the transition propagates along path a-c-e-g.When circuit is fault free, the value of output g is 1 at the required time point t2. Due to the path delay fault along path a-c-e-g, the value of output g remains 0 at t2.

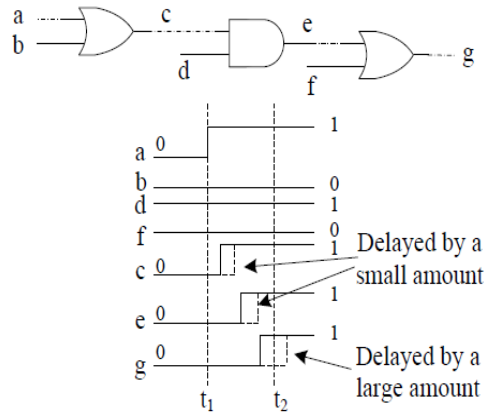


Fig.3.2.path delay fault model

4.2 STUCK-AT FAULT MODEL

Single stuck line is a fault model used in digital circuits. It is used for post manufacturing testing, not design testing. The model assumes one line or node in the digital circuit is stuck at logic high or logic low. When a line is stuck it is called a fault.

Digital circuits can be divided into:

1. Gate level or combinational circuits which contain no storage (latches and/or flip flops) but only gates like NAND, OR, XOR, etc.
2. Sequential circuits which contain storage.

This fault model applies to gate level circuits, or a block of a sequential circuit which can be separated from the storage elements. Ideally a gate-level circuit would be completely tested by applying all possible inputs and checking that they gave the right outputs, but this is completely impractical: an adder to add two 32-bit numbers would require $2^{64} = 1.8*10^{19}$ tests, taking 58 years at 0.1 ns/test. The stuck at fault model assumes that only one input on one gate will be faulty at a time, assuming that if more are faulty, a test that can detect any single fault, should easily find multiple faults.

5.3. GATE EXHAUSTIVE FAULTS

A gate exhaustive test set is defined as a test set that applies all possible input combinations to each gate and observes the gate response at an observation point such as a primary output or a scan cell. The gates can be elementary gates, complex gates, or circuit segments. With, at most, a single bad gate, all Boolean (logical) faults would be detected (unless some gate inputs are redundant) [McCluskey 93]. The effectiveness of a gate exhaustive test set in detecting various kinds of faults such as bridging faults and transistor stuck-on faults is demonstrated by simulation.

Pseudoexhaustive testing partitions a circuit into segments such that the number of inputs of every segment is significantly smaller than the number of primary inputs of the circuit. Exhaustive testing is performed for each segment. However, it is difficult and computationally intensive to find the optimum partitions because the problem is NP-complete [Shperling 87]. In gate exhaustive test set generation, the segment is reduced to each gate in the CUT. The gates can be elementary gates (e.g., AND, OR, NAND, NOR, inverter), complex gates (e.g., XOR, multiplexer, adder, etc.), or circuit segments. Exhaustive input combinations are applied to each gate and the gate response is observed at some observation points. So, the gate exhaustive test metric does not use fault models to generate test patterns.

The circuits in Fig. 1 are used to show examples of nonobservable gate input combinations.

The circuit in Fig. 3.3 (a) is an implementation of a multiplexer, and it is used to show a gate input combination that cannot be applied to the inputs of a gate. In the circuit in Fig. 3.3 (a), the (J1, J2) = (1, 1) combination cannot be applied to the inputs of gate J.

Figure 3.3 (b) is used to show an example of a gate input combination that cannot be sensitized to any observation point. When the (H1, H2) = (0, 0) combination is applied to the inputs of gate H, the output of gate H cannot be sensitized to the output of the circuit (Z) because the sensitization path is blocked by gate J.

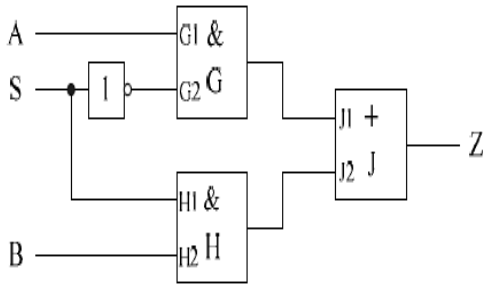


Fig.3.3 an Example circuit to show a gate input combination that cannot be applied to a

gate

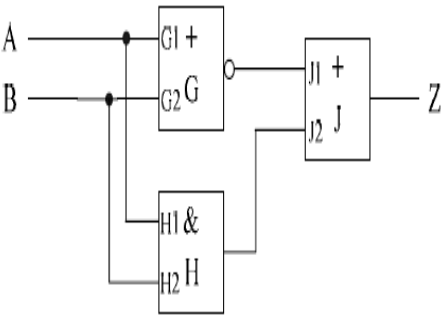


Fig. 3.3.b Example circuit to show a gate input combination that cannot be sensitized to

any observation point

4.4. LFSR

Linear-Feedback Shift Register(LFSR) is a shift register whose input bit is a linear function of its previous state. The most commonly used linear function of single bits is exclusive-or (XOR). Efficient design for Test Pattern Generators & Output Response Analyzers (also used in CRC) FFs plus a few XOR gates better than counter • Fewer gates • Higher clock frequency • Two types of LFSRs External Feedback, Internal Feedback • Higher clock frequency An LFSR generates periodic sequence must start in a non-zero state, The maximum length of an LFSR sequence is 2n -1 does not generate all 0s pattern (gets stuck in that state)The characteristic polynomial of an LFSR generating maximum length sequence is a primitive polynomial A maximum-length sequence is pseudo-random: number of 1s =number of 0s + 1 same number of runs of consecutive 0s and 1s 1/2 of the runs have length1 1/4 of the runs have length 2 (as long as fractions result in integral numbers of runs.

4.5. ON-CHIP TEST GENERATION

The scan configuration assumed in this paper has n scan chains. For simplicity, it is assumed that all the scan chains have the same length, l. This can be achieved by padding each scan chain until its length reaches l. Padding is not needed for the physical circuit, only for the model used by the software procedure to compute the test data for the on-chip test generation logic.

TABLE I
EXAMPLE TEST SET

(a) Scan Vectors

p	0	1	2	3	4	5
s_p	011	110	000	101	010	001

(b) Test Set

j	π_j	j_0	j_1	j_2	$t_{j,0}$	$t_{j,1}$	$t_{j,2}$
0	0	4	1	5	010	110	001
1	0	1	1	3	110	110	101
2	0	2	1	5	000	110	001
3	0	2	2	0	000	000	011
4	1	5	4	5	001	010	001
5	1	5	5	5	001	001	001

One component of the test data stored on-chip is a set of scan vectors, Si = {s0, s1, ..., sv−1}. For illustration, Table I(a) shows the set S0 obtained for a circuit with n = 3 scan chains of length

l = 3. A test tj is formed by selecting n scan vectors from Si, one for every scan chain. With scan vectors sj0 , sj1 , ..., sjn−1, we have that tj = _sj0, sj1 , ..., sjn−1 . For 0 ≤ k < n, the scan vector for scan chain k is denoted by tj,k = sjk .

Table I(b) shows a test set T0 obtained for the circuit from Table I(a). For every test tj , Table I(b) shows the indices j0, j1, j2, and the subtests tj,0, tj,1 and tj,2. The variable πj is explained next. The on-chip test generation logic applies two types of tests. A random test tj = _sj0, sj1 , ..., sjn−1 is designated by πj = 0. In this case, for 0 ≤ k < n, the scan vector sjk for the subtest tj,k is selected randomly. A deterministic test tj = _sj0, sj1 , ..., sjn−1 is designated by πj = 1. In this case, the set of indices j0, j1, ..., jn−1 is stored in an on-chip memory.

The on-chip test generation logic is illustrated by Figure 3.5. The lower part of Figure 2 shows the memory storing the set of scan vectors Si. The size of the memory is v×l. It also shows scan chain k of length l. A multiplexer called MUX2k selects one of the scan vectors from Si depending on the variable called addrk. The number of bits in addrk is log2(v). The selected scan vector is scanned into scan chain k. For a test tj it defines the subtest tj,k. The value of addrk is computed by the logic in the upper part of Figure 3.5. If only random tests are applied, the dashed part of Figure 3.5 is not needed, and an LFSR produces the values for addrk. In general, the deterministic part of the test set Ti, Ti,dtrm, is stored in a memory of size d × n × log2(v), where d is the number of deterministic tests. The entries inside Ti,dtrm in Figure 3.5 correspond

to scan chain k. These are indices of scan vectors that need to be loaded into scan chain k under the deterministic tests t0, t1, ..., td−1. A counter denoted by cnt determines which test is applied through a multiplexer denoted by MUX1k. A count value between 0 and d−1 corresponds to a deterministic test from Ti,dtrm. A count value of d corresponds to a random test. In this case, the LFSR provides log2(v) bits selecting a scan vector randomly from Si. The counter stays at d to apply R random tests, for a parameter R. The memories Si and Ti,dtrm, as well as the counter and LFSR, are common to all the scan chains. In the case of the LFSR, each scan chain uses a distinct subset of bits to obtain a different random number. In addition, each scan chain requires two multiplexers. The overall storage requirements for the two memories are v · l + d · n · log2(v) bits. The memories dominate the size of the on-chip test generation logic.

The entire test generation logic, including both multiplexers for every scan chain, resides close to the memories, within the outline marked TGL in Figure 3.5. Each scan chain is driven by a single line, represented by the output of MUX2k in Figure 3.5. The routing overhead is similar to that of test data decompression logic that drives all the scan chains. For the output response it is assumed that sequential output compaction will be performed by output compaction logic such as a multiple-input shift-register (MISR) [1].

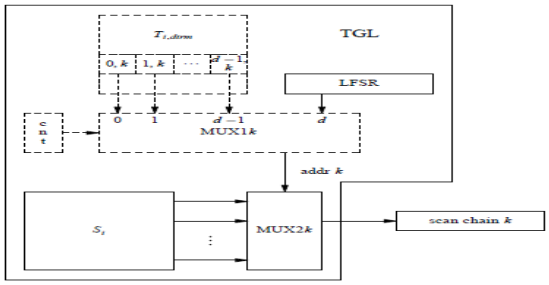


Fig. 3.5. On-chip test generation logic.

5. PROCEDURE FOR COMPUTING STORED TEST DATA

This section describes an iterative software procedure for computing sets of scan vectors Si and deterministic tests Ti,dtrm for on-chip test generation. For simplicity of discussion, Si is associated with a test set Ti that consists of both random and deterministic tests. The non-random tests in Ti define Ti,dtrm.

A. Overview

The procedure accepts a test set Tsa for single stuck-at faults, and a test set Tgexh for gate-exhaustive faults. It produces sets of scan vectors S0, S1, ..., Sm−1, with test sets T0, T1, ..., Tm−1. At the beginning of iteration i = 0, the procedure initializes S0 based on Tsa, as follows. All the distinct scan vectors of Tsa are included in S0, and T0 = Tsa initially. With this initialization, every test in T0 can be expressed in terms of scan vectors from S0. At the beginning of iteration i > 0, Si = Si−1 and Ti = Ti−1. At the end of iteration 0 ≤ i ≤ m − 2, the procedure adds at least one scan vector to Si. It adds tests that use the new scan vectors to Ti. The procedure terminates when, at the end of iteration m−1, it does not add any scan vectors to Sm−1. An arbitrary iteration i proceeds as described next. The procedure referred to as Procedure 0 is applied first to remove unnecessary scan vectors from Si. The test set Ti is modified to ensure that it uses only scan vectors from Si. The procedure referred to as Procedure 1 first stores the current test set Ti in a test set denoted by Tcand, and initializes Ti to be empty. All the target single stuck-at faults are included in Fsa, and all the target gate-exhaustive faults are included in Fgexh. For a parameter R, Procedure 1 includes R random tests in Ti. For every test it performs fault simulation with fault dropping of Fsa and Fgexh. It then uses deterministic tests from Tcand to detect additional faults from Fsa and Fgexh using only scan vectors from Si. All the detectable stuck-at faults from Fsa are guaranteed to be detected by Ti after Procedure 1 is applied. The

procedure terminates if all the detectable gate-exhaustive faults from Fgexh are detected by Ti as well. Otherwise, Procedure 2 adds to Ti a limited number of additional deterministic tests based on Tgexh to detect additional gate-exhaustive faults. As much as possible, Procedure 2 uses only

scan vectors that are already included in Si. When this is not possible, Procedure 2 adds to Ti tests that require new scan vectors, which are added to Si. The procedure prefers tests that require the smallest possible numbers of new scan vectors.

As Ti is modified, the procedure maintains fault detection information for Ti. Fault detection information is obtained by fault simulation with fault dropping of Fsa ∪ Fgexh under Ti. For a fault f ∈ Fsa ∪ Fgexh, the first test in Ti that detects it is denoted by tdet(f). For an undetected fault, det(f) = −1. Fault simulation with fault dropping is also applied at the end of every procedure.

B. Procedure 0

For Procedure 0 all the tests in Ti are considered as deterministic. Procedure 0 associates with every scan vector sp ∈ Si the number of times it is used by a test in Ti. This number is denoted by a(sp). The procedure considers the scan vectors by increasing order of a(sp). This is based on the expectation that it will be easier to remove from Si a scan vector that is used fewer times by tests from Ti. The procedure maintains the variables a(sp) up-to-date as it modifies the test set. A scan vector sp with a(sp) = 0 is removed from Si. When the procedure considers sp for removal, let the subset of tests where sp appears be T (sp). Let the subset of faults that the tests in T (sp) detect be F(sp). To facilitate the modification of the tests in T (sp), the procedure simulates F(sp) under Ti \ T (sp). If a fault f ∈ F(sp) is detected by a test tj ∈ Ti \ T (sp), the procedure assigns det(f) = j, and removes the fault from F(sp). The procedure considers every subtest tj,k such that tj,k = sp separately. For tj,k, it considers as an alternative every scan vector sq ∈ Si such that a(sq) = 0 and q = p. To check whether sq is an acceptable alternative, the procedure assigns tj,k = sq. It then simulates every fault f ∈ Fsa ∪ Fgexh such that det(f) = j under the modified tj . The modification of tj is accepted if all the faults with det(f) = j are detected. Otherwise, the procedure reassigns tj,k = sp. If none of the alternatives to sp is acceptable, the procedure concludes that sp cannot be removed, and it does not consider other subtests where sp appears.

The procedure considers the tests in T (sp) by decreasing order of the number of faults they detect. This is based on the expectation that tests with larger numbers of detected faults are more difficult to modify. If such a test cannot be modified, the procedure will not consider other tests in T (sp). The procedure considers the replacement scan vectors sq by increasing order of the number of appearances in Ti, a(sq). This is based on the expectation that a more uniform use of scan vectors allows faults to be detected more uniformly, and makes it easier to modify the test set. After considering all the scan vectors in Si for removal, if any scan vector was removed from Si, the procedure performs another pass over the scan vectors in Si, in case additional scan vectors can be removed after modifying the test set. Procedure 0 terminates after a pass that does not reduce the number of scan vectors in Si.

C. Procedure 1

Procedure 1 initially assigns Tcand = Ti, and includes all the target faults in Fsa and Fgexh.Procedure 1 includes R random tests in Ti, for a parameter R. A random test tj is constructed by selecting tj,k randomly from Si (or using the LFSR in Figure 2), for 0 ≤ k < n. The procedure performs fault simulation with fault dropping of tj under FsaUFgexh, for every random test tj . Next, the procedure uses tests from Tcand as deterministic tests to detect additional faults from Fsa ∪ Fgexh. The goal is to ensure that both the stuck-at and gate-exhaustive fault coverages of Ti are not lower than those of Tcand. This ensures that the stuck-at

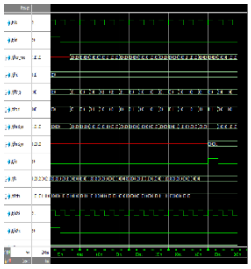
fault coverage is maintained, and the gate-exhaustive fault coverage increases monotonically with every iteration until all the detectable gate-exhaustive faults are detected in the last iteration. For every test tj ∈ Tcand, the procedure applies the following steps. The procedure simulates Fsa ∪ Fgexh under tj . If no faults are detected, the procedure does not consider tj further. Otherwise, it attempts to modify tj to increase the number of faults it detects out of Fsa ∪ Fgexh. For this purpose, the procedure includes in F(tj) all the faults that tj detects. It then considers every scan chain 0 ≤ k < n, and every scan vector sp ∈ Si. If tj,k = sp, the procedure defines a test tmod j that is equal to tj , except that tmod j,k = sp. It simulates the faults in F(tj) under tmod j,k . If all the faults are detected, it also simulates the faults in Fsa ∪Fgexh under tj . If the number of faults detected by tj was increased, the procedure performs another pass over all the scan chains and all the scan vectors in Si. The final test tj is added to Ti, and the faults it detects are removed from Fsa and Fgexh. After

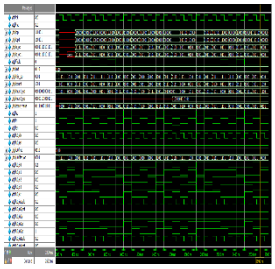
considering every test tj ∈ Tcand, Ti detects all the faults from Fsa ∪ Fgexh that Tcand detects, and possibly additional faults from Fgexh.

D. Procedure 2

Procedure 2 adds to Ti a limited number of tests based on Tgexh. Its goal in selecting which tests will be added is to detect as many additional gate-exhaustive faults as possible using only scan vectors that are already included in Si, or require the addition of as few new scan vectors to Si as possible. It stops after a limited increase in the gate-exhaustive fault efficiency is achieved to avoid a large increase in the storage requirements in a single iteration. The number of tests from Tgexh that the procedure uses depends on the circuit and the iteration. The tests are modified as described below to ensure that as few new scan vectors as possible are added to Si. Procedure 2 is applied iteratively until it achieves its goal. In each pass of Procedure 2, it considers every test tj ∈ Tgexh. When it considers tj , it first performs fault simulation of Fgexh under tj.

6. RESULTS AND DISCUSSIONS





The software procedure for computing the sets Si and Ti was applied to benchmark circuits. The following parameter values were used. For a circuit with K flip-flops, the flip-flops were partitioned into n scan chains such that n2 ≥ K. If necessary, n2 − K flip-flops were added for padding. The length of a scan chain was l = n. This yields a large number of short scan chains. Other configurations with the same property can be used instead.

7. CONCLUSION

This paper described a BIST approach that stores test data onchip, and uses the stored test data to generate both random and deterministic tests on-chip. This approach offers a tradeoff between the amount of stored test data and the comprehensiveness of the test set that can be applied. The paper explored this tradeoff in a specific context where the circuit under consideration has a large number of short scan chains, allowing storage of scan vectors. The initial stored test data is based on a stuck-at test set, but the set of target faults includes single-cycle gate-exhaustive faults.

8.REFERENCES

[1] P. H. Bardell, W. H. McAnney and J. Savir, Built−In T est f or V LSI P seudorandom T echniques, Wiley Interscience, 1987.

[2] I. Pomeranz and S. M. Reddy, ”A Storage Based Built-In Test Pattern Generation Method for Scan Circuits Based on Partitioning and Reduction of a Precomputed Test Set”, IEEE Trans. on Computers, Nov. 2002, pp. 1282-1993.

[3] S. Pateras, ”Security vs. Test Quality: Fully Embedded Test Approaches are the Key to Having Both”, Intl. Test Conf., 2004, Panel P2.2, p. 1413.

[4] D. Xiang, M. Chen and H. Fujiwara, ”Using Weighted Scan Enable Signals to Improve Test Effectiveness of Scan-Based BIST”, in IEEE Trans. on Computers, Dec. 2007, Vol. 56, No. 12, pp. 1619-1628.

[5] R. S. Oliveira, J. Semiao, I. C. Teixeira, M. B. Santos and J. P. Teixeira, ”On-line BIST for Performance Failure Prediction under Aging effects in Automotive Safety-critical Applications”, in Proc. Latin American Test Workshop, 2011, pp. 1-6.

[6] F. Reimann, M. Glas, J. Teich, A. Cook, L. Rodrguez Gmez, D. Ull, H.- J. Wunderlich, P. Engelke and U. Abelein, ”Advanced Diagnosis: SBST and BIST Integration in Automotive E/E Architectures”, in Proc. Design Autom. Conf., 2014, pp. 1-9.

[7] S. U. Hussain, S. Yellapantula, M. Majzoobi and F. Koushanfar, ”BISTPUF: Online, Hardware-Based Evaluation of Physically Unclonable Circuit Identifiers”, in Proc. Intl. Conf. on Computer-Aided Design, 2014, pp. 162-169.

[8] R. Wang, K. Chakrabarty and S. Bhawmik, ”Built-In Self-Test and Test Scheduling for Interposer-Based 2.5D IC”, ACM Trans. on Design Automation, Vol. 20, No. 4, Sep. 2015, Art. 58.

[9] D. Xiang, X. Wen and L. Wang, ”Low-Power Scan-Based Built-In SelfTest Based on Weighted Pseudorandom Test Pattern Generation and Reseeding”, in IEEE Trans. on VLSI Systems, March 2017, Vol. 25, No. 3, pp. 942-953.

[10] O. E. Erol and S. Ozev, ”Knowledge- and Simulation-Based Synthesis of Area-Efficient Passive Loop Filter Incremental Zoom-

ADC for BuiltIn Self-Test Applications", ACM Trans. on Design Automation, Vol. 24, No. 1, Jan. 2019, Art. 3.

[11] A. H. Baba and S. Mitra, "Testing for Transistor Aging", in Proc. VLSI Test Symposium, 2009, pp. 215-220.

[12] Y. Sato, "Circuit Failure Prediction by Field Test - A New Task of Testing", in Proc. Intl. Symp. on Defect and Fault Tolerance in VLSI Systems, 2010, pp. 69-70.

[13] S. Jin, Y. Han, H. Li and X. Li, "Unified Capture Scheme for Small Delay Defect Detection and Aging Prediction", IEEE Trans. on VLSI Systems, May 2013, Vol. 21, No. 5, pp. 821-833.

[14] A. Sivadasan, R. J. Shah, V. Huard, F. Cacho and L. Anghel, "NBTI Aged Cell Rejuvenation with Back Biasing and Resulting Critical Path Reordering for Digital Circuits in 28nm FDSOI", in Proc. Design, Automation & Test in Europe Conf., 2018, pp. 997-998.

[15] A. Jas, C. V. Krishna and N. A. Touba, "Weighted Pseudorandom Hybrid BIST", IEEE Trans. on VLSI Systems, Dec. 2004, Vol. 12, No. 12, pp. 1277-1283.

[16] S. Wang, K. J. Balakrishnan and S. T. Chakradhar, "XWRC: ExternallyLoaded Weighted Random Pattern Testing for Input Test Data Compression", Intl. Test Conf., 2005, Paper 24.2, pp. 1-10.

[17] M. Filipek, G. Mrugalski, N. Mukherjee, B. Nadeau-Dostie, J. Rajski, J. Solecki and J. Tyszer, "Low-Power Programmable PRPG With Test Compression Capabilities", IEEE Trans. on VLSI Systems, June 2015, Vol. 23, No. 6, pp. 1063-1076.