ERROR MASKING IN SELF-TESTABLE CIRCUITS

Albrecht P. Stroele, Hans-Joachim Wunderlich

University of Karlsruhe
Institute of Computer Design and Fault Tolerance
(Prof. Dr. D. Schmid)
P. O. Box 6980, D-7500 Karlsruhe 1
Federal Republic of Germany

ABSTRACT:

In a self-test environment signature analysis is used to compact the test responses. In large circuits the test execution is divided into a number of subtasks each producing a signature in a self-test register. Aliasing occurs, if a faulty response sequence leads to a correct signature in a signature register. Aliasing probabilities for single signature registers are widely investigated.

In this paper the effects of error masking in a multitude of signature registers are analysed. It is shown that a self-test can always be scheduled such that evaluating signatures only at the end of the complete test execution is sufficient. A method is presented to compute the probability that a fault leads to at least one faulty signature in a set of self-test registers. This method allows the computation of the fault coverage with respect to the complete test execution. A minimal subset of all self-test registers can be selected, so that only the signatures of these self-test registers have to be evaluated and the fault coverage is almost not affected.

The benefits of this approach are a smaller number of self-test registers in the scan path, a smaller number of signatures to be evaluated, a simplified test control unit, and hence a significant reduction of the hardware required for built-in self-test structures.

Key words: Built-in self-test, error masking, fault coverage, signature analysis.

1. INTRODUCTION

With increasing circuit complexity, self-test using pseudo-random or (pseudo-)exhaustive test patterns is receiving more and more attention and its importance is still growing due to its well-known advantages [12]. One or more test pattern generators (TPGs), e.g. linear feedback shift registers (LFSRs), produce patterns that are applied to the inputs of the circuit under test (CUT). The test responses of the CUT are fed to a multiple input signature register (MISR) and compressed into a signature (see figure 1). By comparing the signature obtained for the fault-free circuit, it is decided if the tested circuit is faulty. However, even false test responses may result in the correct signature. This is called error masking or aliasing.

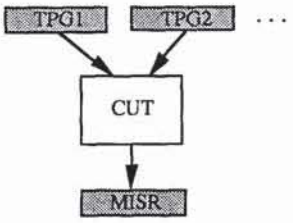

Figure 1: General test configuration (one test unit)

Larger and more complicated circuits cannot be dealt with in this manner, but they can be made self-testable using multi-mode self-test registers (STRs) like the well-known BILBO (built-in logic block observer [8]) or GURT (generator of unequiprobable random tests [16]). In the test mode STRs generate patterns or perform signature analysis. By an appropriate placement of these STRs the circuit is subdivided into segments that are completely bounded by STRs as in figure 1. These segments are called *test units*. Each test unit contains exactly one STR, that is configured as an MISR when the segment is tested, and a set of STRs that generate test patterns for all the inputs of this segment. A test unit can be tested independently from the rest of the circuit. Thus the test of the whole circuit (*global test*) consists of processing all the test units. But up to now investigations of the aliasing probability in signature analysis only deal with a single signature register, as it is used within one single test unit (see [3], [4], [7], [13], [14], and others).

To reduce the time required for the global test one tries to process as many test units in parallel as possible. The problem of test scheduling is to organize the execution of all the single test units so that the available resources are optimally utilized. Scheduling algorithms are presented in e.g. [2]. In order to obtain a completely self-testable circuit, the test schedule must be implemented by a test control unit. It is a common practice to read out the contents of the STRs and evaluate the signature,

every time the test of a test unit has been completed. This usually implies that all STRs must be integrated into a scan path. In addition, the task of the test control unit becomes quite complex.

In this paper we discuss some conditions for the test registers and the test schedule, which allow the contents of the STRs to be read out just once, when the global test is completed. Furthermore, it is shown that only the signatures in a subset of all STRs have to be evaluated, in order to get sufficient fault coverage and a low fault masking probability. This yields important advantages:

- Shorter scan path (only a subset of the set of STRs is integrated into the scan path)
- Fewer signatures to evaluate
- Simple test control

All three points contribute to a significant reduction of the additional hardware needed for a built-in self-test.

To achieve these results, we analyse the effects that a faulty signature, occuring at some time during the test, can have in the further course of the test execution. In particular, we investigate how faulty signatures are propagated through a circuit. We present a method for computing the probabilities of faulty signatures in the STRs in these situations. We assume that each test unit is processed exactly once (without interruption). Then each STR is used as an MISR just once in the test schedule. This restriction eases clarity of presentation and the extension to the general case is straightforward.

In section 2, we analyse what effects a faulty signature can have during test execution. Section 3 establishes a formalization of the problem by means of a system of boolean equations. There is a unique solution for this system of equations and a method for computing the probabilities of faulty signatures is presented in section 4. Section 5 extends these ideas to the computation of fault coverage and the selection of a minimal set of STRs whose signatures have to be evaluated. Section 6 demonstrates the method with an example and some simulation results that confirm the computed values. In section 7, some extensions of the presented method and subjects of further research are briefly discussed. The paper concludes with a summary in section 8.

2. PROPAGATION OF FAULTY SIGNATURES

Generally an LFSR with primitive feedback polynomial is used as signature register. With increasing test length the aliasing probability asymptotically converges to the value 2^{-k}, where k is the length of the LFSR that is used as signature register [14]. The authors of that paper assume that consecutive bit errors (differences from the fault-free case) at the inputs of the signature register are statistically independent. However, dependences may exist between the bit errors, simultaneously occuring at different inputs of the signature register. This result is valid not only for LFSRs, but also for cellular automata with linear state transition functions.

If a fault is detectable by the generated pattern sequences, in at least one test unit a bit error will occur at the inputs of the signature register. Without aliasing, a faulty signature will be produced in this signature register. Let us assume that this STR is not initialized again. If the STR with incorrect contents is used in one of the following test units to generate a test pattern sequence, this sequence will not be the same as the sequence produced with the correct starting value. In this case the STR will retain a faulty signature and with a high probability (see section 4) the new pattern sequence will cause a faulty signature in the signature register of the second test unit, too. In this way faulty signatures may propagate through the circuit.

Figure 2 illustrates this with an example. The combinational logic block CLB_1 is tested first, then CLB_2. The combinational logic block CLB_1 contains a fault. S_1, S_2, S_3, and S_4 are self-test registers. Provided that error masking does not occur, a faulty signature will be in S_1 after processing the first test unit (test of CLB_1), and after the second test unit (test of CLB_2, using S_1 and S_4 as test pattern generators) a faulty signature will be in S_2, too, although there is no fault in CLB_2.

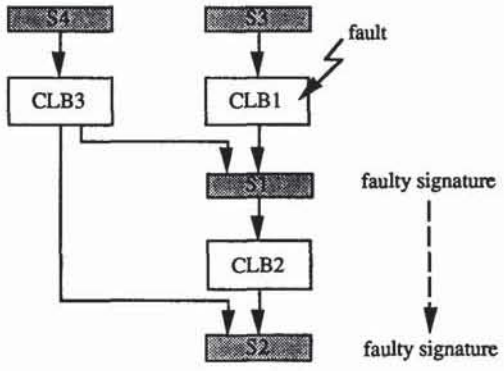

Figure 2: Propagation of faulty signatures

A faulty signature that occurs during test execution may cause further faulty signatures, even in parts of the circuit that are fault-free. Faulty signatures can propagate only in the direction of the data flow. Furthermore, the propagation depends on the test schedule that specifies the succession of the test units when the test is executed.

In the subsequent sections we make the following assumptions:

a) All STRs are initialized correctly before test execution begins. (The extension to the general case is straightforward.)

b) When an STR is operating in signature analysis mode its contents are not used at the same time as test patterns.

c) When an STR is operating in pattern generation mode its function is not affected by the incoming data sequence.

d) No faulty signature is lost by clocking the STR in normal mode (system mode) or by initializing it at an inappropriate time. (The extension to the general case is straightforward.)

e) Every STR is scheduled only once for signature analysis.

The conditions b) and c) ensure that the generated patterns do not depend on the circuit function and are pseudo-random or pseudo-exhaustive.

After the processing of a test unit an STR is in one of two different states:

State "0": The contents of the STR correspond to that of the fault-free case.

State "1": The contents of the STR differ from that of the fault-free case, i.e. the STR contains a faulty signature.

The state of an STR operating in TPG-mode will remain unchanged. A correct/false starting value will always lead to a correct/false final value (last pattern of the generated pattern sequence). This is true for all self-test registers based on LFSRs and cellular automata with linear state transition functions. However, when an STR is operating as a signature register, its state may change if the processed test unit contains a detectable fault or at least one of the involved pattern generating STRs has got an incorrect starting value and consequently produces a different pattern sequence. Due to conditions d) and e), the state of the STR does not change any more afterwards. Thus the probability for error masking is the same if the signature is read immediately after the signature analysis phase and after the completion of the global test, respectively.

3. MODELING AND FORMAL REPRESENTATION

In this section the generation and propagation of faulty signatures (incorrect STR-contents) during test execution is modeled by means of a graph. Then a method is presented for generating a system of boolean equations to describe the conditions for the occurrence of faulty signatures. We are using the following notations:

Test units: $u_1, u_2, ..., u_m$
Self-test registers (STRs): $S_1, S_2, ..., S_n, \quad n \geq m$
Length of self-test register S_i: k_i

Self-test registers used in test unit u_i:
$TPG(u_i) = \{ S_g \mid S_g$ is used for test pattern generation in test unit $u_i \}$,
S_i is used for signature analysis in test unit u_i.

(Primary inputs and primary outputs are treated in the same way as STRs.)

3.1 Graph representation

For a self-test configuration which satisfies the conditions a) to e), a graph can be constructed for modeling the generation and propagation of faulty signatures, if the following informations are given:

- Circuit structure (including built-in STRs)
- Test schedule as a ordered set of test units $\{u_1, u_2, ..., u_m\}$
- Location of a single or multiple fault F in the circuit, represented as a set of test units $U_F = \{ u_i \mid F$ is located in test unit $u_i \}$

The paths of the *signature graph* $G = (V, E)$ model how faulty signatures caused by the fault F can propagate to other STRs while the test units are processed. The graph contains nodes $v_{F,i} \in V$ representing the fault F in test unit u_i and nodes $v_i \in V$, representing STRs S_i, that can get incorrect contents. STRs that can never get incorrect contents during test execution do not appear in the signature graph. If the fault F is located in test unit u_i and thus may cause STR S_i to get incorrect contents, the corresponding nodes of the graph are connected by a directed edge $(v_{F,i}, v_i) \in E$. Similarly if an STR S_i is operating in TPG-mode with an incorrect starting value and thus can cause another STR S_j to get incorrect contents, the corresponding nodes are connected by a directed edge $(v_i, v_j) \in E$.

The signature graph G is formally described by:

$$V = \{ v_{F,i} \mid u_i \in U_F \} \cup \{ v_i \mid u_i \in U_F \vee \exists g \, [v_g \in V \wedge S_g \in TPG(u_i)] \}$$

$$E = \{ (v_{F,i}, v_i) \mid u_i \in U_F \} \cup \{ (v_g, v_i) \mid v_g \in V \wedge S_g \in TPG(u_i) \}$$

We clarify this definition by using the matrix multiplication circuit of figure 3 as an example [5]. The circuit contains combinational logic blocks (CLB), pipeline-structured sequential logic blocks, and self-test registers (S_i).

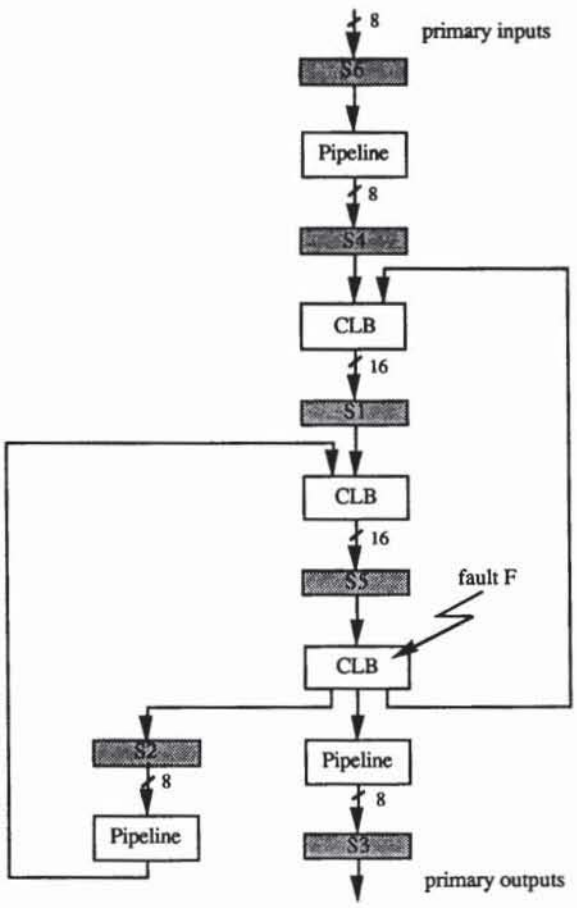

Figure 3: Circuit for matrix multiplication with built-in self-test registers S_i

Test unit	pattern generating register(s)	signature register	starting time	terminating time
u_1	S_4, S_5	S_1	0	t_1
u_2	S_5	S_2	0	t_1
u_3	S_5	S_3	0	t_1
u_4	S_6	S_4	t_1	t_2
u_5	S_1, S_2	S_5	t_1	t_2

Table 1: Test units for the multiplication circuit of figure 3
(t_1: termination of test units u_1, u_2, u_3
t_2: termination of test units u_4, u_5)

The global test is composed of the test units listed in table 1. First, test units u_1, u_2, and u_3 are processed, then test units u_4 and u_5. The fault F marked in figure 3 is located in the set of test units $U_F = \{u_1, u_2, u_3\}$.

The circuit structure of figure 3, the test schedule of table 1, and the marked fault F, lead to a signature graph as shown in figure 4.

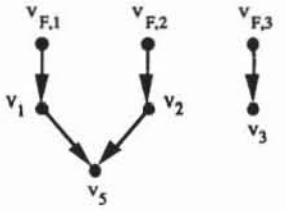

Figure 4: Signature graph G for the multiplication circuit of figure 3

The signature graph can be automatically constructed by the algorithm of appendix A. This algorithm guarantees that the signature graph does not contain any cycles.

3.2 System of boolean equations

Each node v_i of the graph G corresponds to a test unit u_i whose processing can result in a faulty signature in STR S_i. To describe the conditions for the generation and propagation of faulty signatures we use the following boolean variables with values 0 (false) and 1 (true):

b_i : STR S_i (used in test unit u_i as signature register) contains a faulty signature after the processing of test unit u_i.

$b_{F,i}$: The fault F is located in test unit u_i and can be observed at the inputs of signature register S_i, when test unit u_i is processed.

c_i : Provided that there were bit errors at the inputs of signature register S_i, then signature register S_i contains a faulty signature after the processing of test unit u_i. (c_i models aliasing.)

For each node v_i of the graph G we set up one equation

$$b_i = \Big(\bigvee_{j \in M_{TPG}(v_i)} b_j \vee b_{F,i} \Big) \wedge c_i \qquad \text{if } v_{F,i} \in V \qquad (1)$$

and

$$b_i = \Big(\bigvee_{j \in M_{TPG}(v_i)} b_j \Big) \wedge c_i \qquad \text{if } v_{F,i} \notin V$$

where $M_{TPG}(v_i) = \{g \mid (v_g, v_i) \in E\}$

The equation characterizes the situation in test unit u_i. If at least one of the pattern generating STRs starts with incorrect contents or the fault F is detectable in this test unit, then bit errors occur at the inputs of STR S_i and this STR may get incorrect contents.

4. COMPUTING THE PROBABILITIES OF FAULTY SIGNATURES

To determine the probability that a fault F causes a faulty signature in a STR S_i during the the global test, we have to compute the probability that the boolean variable b_i in the system of boolean equations gets the value 1. We assume that the test length for each test unit is long enough for the transient part of the aliasing probability to die down so that the steady state value of 2^{-k_i} for the aliasing probability is sufficiently exact [14]. The calculations of [7] and the simulation results of [13] and [14] show that the aliasing probability converges fast to its steady state when a LFSR with primitive feedback polynomial is used. In general this condition holds as the test lengths increase with the reciprocal of the lowest detection probabilities ([11], [17]).

The system of boolean equations is transformed into a system of arithmetic equations by using the following substitutions:

Boolean constant $b_{F,i}$ → $p_{F,i} = P(b_{F,i}=1)$:
probability of detecting the fault F at the inputs of signature register S_i, while test unit u_i is processed (e.g. estimated by PROTEST [15])

Boolean constant c_i → $P(c_i=1) = 1 - 2^{-k_i}$

Boolean variable b_i → $p_i = P(b_i=1)$:
probability of a faulty signature in STR S_i, after test unit u_i has been processed

In [10] a method for creating the system of arithmetic equations was proposed and in [1] an operator ⊗ was introduced as an abbreviation. The operator ⊗ takes into account the dependences among events.

Using these notations, the boolean operators are also substituted:

$$\neg b_i \rightarrow 1 - p_i$$

$$b_g \wedge b_i \rightarrow p_g \otimes p_i$$

$$b_g \vee b_i \rightarrow 1 - (1 - p_g) \otimes (1 - p_i)$$

Let be $M_{TPG}(v_i) := \{ g \mid (v_g, v_i) \in E \} = \{ g_1, g_2, ..., g_r \}$. Then the arithmetic equations corresponding to the boolean equations (1) are

$$p_i = [1 - (1 - p_{g_1}) \otimes ... \otimes (1 - p_{g_r}) \otimes (1 - p_{F,i})] \otimes (1 - 2^{-k_i}) \quad \text{if } v_{F,i} \in V \tag{2}$$

and

$$p_i = [1 - (1 - p_{g_1}) \otimes ... \otimes (1 - p_{g_r})] \otimes (1 - 2^{-k_i}) \quad \text{if } v_{F,i} \notin V$$

The solution of this system of arithmetic equations can now be determined by a procedure for computing signal probabilities as described in e.g. [10], [15], [1]. Since the signature graph does not contain any cycles, and the signal probabilities of all inputs are known, each variable can be expressed as a sum of products containing only known quantities. So the solution is unique.

The procedure described in appendix B gives an exact solution. It is well-known that an exact computation of this kind is an NP-hard problem. But test planning and scheduling aim at a moderate number of STRs. The number of test units and the number of reconvergent fanout stems in G are moderate, too. Hence the size of the problem is quite small, and an exact computation is feasible.

5. SELECTING AN OPTIMAL SET OF SIGNATURES

In sections 3 and 4 a method was shown for computing the probability that a single fault F leads to a faulty signature in a specific STR. In order to determine the probability that a single fault leads to at least one faulty signature in a set S of STRs, a new equation

$$b_{F\,\text{detected}} = \left(\bigvee_{j \in A} b_j \right) \tag{3}$$

$$\text{where} \quad A = \{ g \mid S_g \in S \wedge v_g \in V \}$$

is added to the system of boolean equations, and the probability $p_{F\,\text{detected}} = P(b_{F\,\text{detected}} = 1)$ is computed using the procedure of appendix B.

Fault coverage can be defined as the average probability for detecting a fault [9]. Usually error masking is ignored when the fault coverage of a self-test procedure is computed. In order to determine the fault coverage with respect to error masking, the probabilities $p_{F\,\text{detected}}$ for all faults of the circuit are needed. All faults located in the same test unit(s) lead to the same graph G and the same system of equations. Thus it is sufficient to solve the equations just once with symbolic values $p_{F,i}$ and then insert the actual values of $p_{F,i}$ into this solution for all these faults.

Finally we have to choose the set of STRs whose contents (signatures) are read out at the end of the global test. Of course evaluating all the signatures yields maximum fault coverage, but

many signatures can be omitted without a significant impact on the fault coverage. Using the method presented above it is possible to select a minimal set of STRs, such that the fault coverage does not drop below a given value. Certainly this set must include the STRs whose signatures cannot propagate to other STRs. Then the STRs that are most affected by error masking are added to the set, until the resulting fault coverage is sufficient. All other STRs can be implemented with less hardware because they do not have to be readable. The test control unit becomes simpler and also needs less hardware, since the number of signatures to evaluate is reduced. The next section will demonstrate these benefits.

6. RESULTS

The analysis of error masking probabilities will be demonstrated using the example of figure 3. According to section 3.2 we get the following system of boolean equations:

$$\begin{aligned}
&\text{(i)} && b_1 = b_{F,1} \wedge c_1 \\
&\text{(ii)} && b_2 = b_{F,2} \wedge c_2 \\
&\text{(iii)} && b_3 = b_{F,3} \wedge c_3 \\
&\text{(iv)} && b_5 = (b_1 \vee b_2) \wedge c_5
\end{aligned}$$

With the probabilities of faulty signatures

$$\begin{aligned}
P(c_1{=}1) &= P(c_5{=}1) = 1 - 2^{-16} = 0.999985 \\
P(c_2{=}1) &= P(c_3{=}1) = 1 - 2^{-8} = 0.996094
\end{aligned}$$

a system of arithmetic equations is obtained (see section 4)

$$\begin{aligned}
&\text{(i)} && p_1 = p_{F,1} \otimes 0.999985 \\
&\text{(ii)} && p_2 = p_{F,2} \otimes 0.996094 \\
&\text{(iii)} && p_3 = p_{F,3} \otimes 0.996094 \\
&\text{(iv)} && p_5 = (p_1 + p_2 - p_1 \otimes p_2) \otimes 0.999985
\end{aligned}$$

and the solution with symbolic values $p_{F,1}, p_{F,2}, p_{F,3}$ is

$$\begin{aligned}
p_1 &= 0.999985 \cdot p_{F,1} \\
p_2 &= 0.996094 \cdot p_{F,2} \\
p_3 &= 0.996094 \cdot p_{F,3} \\
p_5 &= 0.999969 \cdot p_{F,1} \\
&\quad + \; 0.996079 \cdot p_{F,2} \\
&\quad - \; 0.996063 \cdot p_{F,1} \cdot p_{F,2}
\end{aligned}$$

For $p_{F,1} = p_{F,2} = p_{F,3} = 1$ we have

$$\begin{aligned}
p_1 &= 0.999985 \\
p_2 &= 0.996094 \\
p_3 &= 0.996094 \\
p_5 &= 0.999985
\end{aligned}$$

Using the additional equation (3), we can compute the probabilities for a faulty signature in at least one STR of a set of STRs.

In table 2 these values are compared with results of fault simulation for the same circuit. In each of the independent simulation runs a complete execution of the global test was simulated including all test units. After each run it was checked, which STRs contained faulty signatures.

There is no significant difference between the precomputed values and the actually obtained frequency. For several other circuits, especially with complicated reconvergent paths in the graph G, the computed probabilities for faulty signatures were also validated by simulation.

Set of self-test registers	frequency of faulty signatures	computed probability of faulty signatures
S_1	0.999984	0.999985
S_2	0.996106	0.996094
S_3	0.996074	0.996094
S_4	0	0
S_5	0.999985	0.999985
S_6	0	0
S_3, S_5	1.000000	0.999999
S_1, S_2, S_3, S_5	1.000000	0.999999

Table 2: Frequency (10^7 simulation runs) and probabilities of faulty signatures

The results for the multiplication circuit show some remarkable points. Although the fault F is not located in test unit u_5, error masking in STR S_5 is almost the same as error masking in S_1. As a consequence one has the choice to either evaluate the signature of S_1 or S_5. When all potential fault locations in the multiplication circuit are examined in the same way as above (using the same test schedule), it can be shown that it is sufficient to evaluate the signatures in S_3, S_4, and S_5 at the end of the global test. The decrease of the expected fault coverage due to ignoring the signatures in S_1 and S_2 is less than 10^{-5}. Table 2 confirms this for the fault F by simulation.

In this case STRs S_1 and S_2 can be implemented with less hardware, because they do not have to be readable. The routing area required for the scan path is also reduced, since only S_3, S_4, S_5 (and perhaps S_6) have to be integrated into the scan path. With respect to the test control unit we have benefits, too. Firstly, the contents of the STRs are read only once at the end of the global test. Secondly, only the contents of a subset of all STRs are read. And thirdly, fewer reference values for the evaluation of the resulting signatures are needed.

In [6] a microprogrammed control unit is proposed. The parts of this control unit that require most silicon area are the microcode ROM and the constant data ROM. Using our method to simplify

the task of the test control unit, we can reduce the length of the microcode from 17 to 9 words, and the number of words in the constant data ROM from 7 to 3 words. Altogether we can implement a built-in self-test with substantially less test hardware.

The circuit of figure 5 is a more difficult problem for the analysis of error masking, since it contains nested cycles and reconvergent paths. All STRs are 16 bit registers. Table 3 lists the test units, and figure 6 shows the signature graph for the fault F.

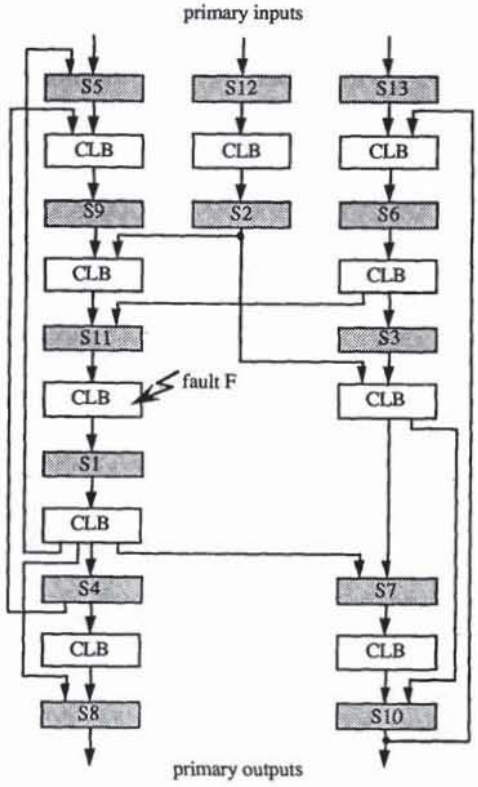

Figure 5: Circuit with built-in self-test registers S_i

Again, if all potential fault locations are considered, it is sufficient to evaluate only the signatures in a small subset of STRs, namely in S_8, S_{10}, and S_{11}. The length of the microcode for the control unit is reduced from 29 to 13 words, and the number of words in the constant data ROM from 9 to 5 words.

Test unit	pattern generating register(s)	signature register	starting time	terminating time
u_1	S_{11}	S_1	0	t_1
u_2	S_{12}	S_2	0	t_1
u_3	S_6	S_3	0	t_1
u_4	S_1	S_4	t_1	t_2
u_5	S_1	S_5	t_1	t_2
u_6	S_{10}, S_{13}	S_6	t_1	t_2
u_7	S_1, S_2, S_3	S_7	t_1	t_2
u_8	S_1, S_4	S_8	t_2	t_3
u_9	S_4, S_5	S_9	t_2	t_3
u_{10}	S_3, S_7	S_{10}	t_2	t_3
u_{11}	S_2, S_6, S_9	S_{11}	t_3	t_4

Table 3: Test units for the circuit of figure 5

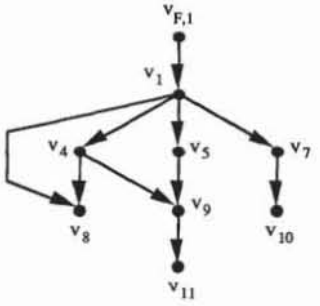

Figure 6: Signature graph G for the circuit of figure 5 and fault F

7. EXTENSIONS AND FURTHER RESEARCH

Looking back at the conditions a) to e) in section 2, the restrictions b) and c) are essential for all self-test strategies using pseudo-random or pseudo-exhaustive patterns. The conditions a) and d) can be dropped, if the signature graph is slightly modified. STRs that are not initialized correctly are handled like faults at the inputs and outputs of the combinational parts under test, respectively. Initializing an STR at some time during test execution means that this STR is forced into the correct state. This is modeled by deleting all edges in the signature graph that start from the node corresponding to the initialized STR and lead to nodes corresponding to STRs that are used as signature registers after that time.

Even condition e) can be dropped. STRs that are repeatedly scheduled for signature analysis are represented in the signature graph by one node for each signature analysis phase. These

nodes are connected by edges. This makes the equations (1), (2), and (3) more complex. But the procedure of appendix B still gives a unique solution.

In this paper we used the steady state aliasing probabilities. These are very good estimates for the random and (pseudo-) exhaustive case. But deterministic test lengths are shorter and then the aliasing probabilities can differ from the steady state. For this reason our approach is not generally applicable to deterministic test strategies. The situation when the steady state is not reached is a subject of further research. Another interesting point not considered so far, is the influence of correlated consecutive bit errors, as they are caused by faulty sequential circuits.

On the other hand, the presented method gives a new optimization objective for test scheduling. The test schedule should minimize the number of signatures which must be evaluated. An obvious heuristic for this goal is a schedule, where the order of the test units u_i corresponds to the data flow. An exact formulation and solution for this problem is currently under research.

8. CONCLUSIONS

The effects of faulty signatures occuring at some time during test execution have been analysed. The generation and the propagation of faulty signatures have been modeled by means of a graph and a system of equations. To determine the probability that a fault leads to at least one faulty signature in a set of STRs, a method has been presented that gives an exact solution. This is the basic instrument to compute the fault coverage with respect to the global test and also to determine a minimal set of STRs whose signatures have to be evaluated after the completion of the global test. The presented method has been demonstrated with an example and validated by simulation.

The benefits of this approach are threefold: fewer signatures have to be evaluated, the number of self-test registers to be integrated into the scan path is smaller, and the test control unit is simplified. In consequence the amount of hardware required for a built-in self-test is substantially reduced.

REFERENCES

[1] E. J. Aas, M. R. Mercer, "Algebraic and Structural Computation of Signal Probability and Fault Detectability in Combinational Circuits", Proc. International Symposium on Fault-Tolerant Computing FTCS-17, pp. 72-77, Pittsburgh 1987

[2] G. L. Craig, C. R. Kime, K. K. Saluja, "Test Scheduling and Control for VLSI Built-In Self-Test", IEEE Transactions on Computers, Vol. 39, No. 9, September 1988, pp. 1099-1109

[3] M. Damiani et al., "Aliasing in Signature Analysis Testing with Multiple-Input Shift-Registers", Proc. 1st European Test Conference ETC-89, pp. 346-353, Paris 1989

[4] S. K. Gupta, D. K. Pradhan, "A New Framework for Designing and Analyzing BIST Techniques: Computation of Exact Aliasing Probability", Proc. International Test Conference ITC-88, Washington, 1988, pp. 329-342

[5] P. Gutberlet, "Entwurf eines schnellen Matrizenmultiplizierers", Studienarbeit Fakultät Informatik, Universität Karlsruhe, 1988

[6] O. F. Haberl, H.-J. Wunderlich, "The Synthesis of Self-Test Control Logic", Proc. COMPEURO-89, pp. 5.134-5.136, Hamburg 1989

[7] A. Ivanov, V. K. Agarwal, "An Iterative Technique for Calculating Aliasing Probability of Linear Feedback Signature Registers", Proc. 18th International Symposium on Fault-Tolerant Computing FTCS-18, pp. 70-75, Tokyo 1988

[8] B. Koenemann, J. Mucha, G. Zwiehoff, "Built-In Logic Block Observation Techniques", Proc. IEEE Test Conference, pp. 37-41, Cherry Hill, New Jersey, 1979

[9] Y. K. Malaiya, S. Yang, "The Coverage Problem for Random Testing", Proc. International Test Conference ITC-84, pp. 237-245, 1984

[10] K. P. Parker, E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks", IEEE Transactions on Computers, Vol. 26, No. 6, June 1973, pp. 668-670

[11] J. Savir, P. H. Bardell, "On Random Pattern Test Length", IEEE Transactions on Computers, Vol. C-33, No. 6, June 1984, pp. 467-474

[12] D. Schmid, H.-J. Wunderlich et al., "Integrated Tools for Automatic Design for Testability", Proc. IFIP WG 10.2 Workshop on Tool Integration and Design Environments, pp. 233-258, Paderborn, FRG, 1987

[13] T. W. Williams, W. Daehn, "Aliasing Errors in Multiple Input Signature Analysis Registers", Proc. 1st European Test Conference ETC-89, pp. 338-345, Paris 1989

[14] T. W. Williams, W. Daehn, "Aliasing Probability for Multiple Input Signature Analysers with Dependent Inputs", Proc. COMPEURO-89, pp. 5.120-5.127, Hamburg 1989

[15] H.-J. Wunderlich, "PROTEST: A Tool for Probabilistic Testability Analysis", Proc. 22nd Design Automation Conference, pp. 204-211, Las Vegas 1985

[16] H.-J. Wunderlich, "Self-Test Using Unequiprobable Random Patterns", Proc. International Symposium on Fault-Tolerant Computing FTCS-17, pp. 258-263, Pittsburgh 1987

[17] H.-J. Wunderlich, "Multiple Distributions for Biased Random Test Patterns", Proc. International Test Conference ITC-88, pp. 236-244, Washington 1988

Appendix A: Algorithm to construct the signature graph of section 3.1

```
begin
V = ∅;
E = ∅;
for i = 1, 2, ..., m
    begin
    /* if a faulty signature may result */
    if (u_i ∈ U_F ∨ ∃g [ v_g ∈ V ∧ S_g ∈ TPG(u_i) ] )
        begin
        /* add new node (signature register) */
        V = V ∪ {v_i};
        if (u_i ∈ U_F)
            begin
            /* add new node (fault) */
            V = V ∪ {v_F,i};
            /* add edge from fault node */
            E = E ∪ {(v_F,i, v_i)}
            end
        for all v_g ∈ {v_h | v_h ∈ V ∧ S_h ∈ TPG(u_i)}
        /* add edges from other nodes (TPGs) */
            E = E ∪ {(v_g, v_i)}
        end
    end
end.
```

Appendix B: Procedure to compute the probabilities of faulty signatures (see section 4)

1) Determine the set of all reconvergent fanout stems in the signature graph G:
   $M_R = \{i \mid v_i \text{ is a reconvergent fanout stem in G}\}$.

2) Order the equations of the system with respect to increasing indices i of the variables p_i on the left side of the equations. (This is the chronological order of the corresponding test units in the test schedule and also agrees with traversing the graph G from the roots to the leaves.)

3) Process all the equations in this order one by one:

   Let Q_j be the next equation. In the order of decreasing indices i substitute all variables p_i with $i \notin M_R$ on the right side of Q_j by the right side of the equation for p_i. After each substitution step:

   - Execute multiplication operations and use distributivity until the right side of the equation consists only of a sum of products (including $\otimes$)
   - collect constant values
   - reduce $p_i \otimes p_i = p_i$ as far as possible using associativity

4) Process all the equations in reversed order (beginning with the last equation) and step by step substitute all variables p_i with $i \in M_R$. The substitutions are done in the same way as in 3).