MAXIMIZING THE FAULT COVERAGE IN COMPLEX CIRCUITS BY MINIMAL NUMBER OF SIGNATURES

Hans-Joachim Wunderlich, Albrecht P. Ströle

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:

Many self-test strategies use signature analysis to compress the test responses. In complex circuits the test execution is divided into a number of subtasks, each producing a signature in a self-test register. Whereas the conventional approach is to evaluate all these signatures, this paper presents methods to minimize the number of evaluated signatures without reducing the fault coverage. This is possible, since the signatures can influence one another during the test execution. For a fixed test schedule a minimal subset of signatures can be selected, and for a predetermined minimal subset of signatures the test schedule can be constructed such that the fault coverage is maximum. Both approaches result in significant hardware savings when a self-test is implemented.

Keywords: BIST, signature analysis, test scheduling.

1. Introduction

In order to implement a built-in self-test often some system registers are augmented to multi-mode *self-test registers* (STRs) like the well-known BILBO (built-in logic block observer [8]) or GURT (generator of unequiprobable random tests [12]). In the test mode, STRs generate patterns or perform signature analysis. By an appropriate placement of the STRs, in the test mode all global feedback loops are cut, and the circuit is subdivided into segments that are completely bounded by STRs (e.g. [2, 9]).

These segments are called *test units* (figure 1). A test unit contains a set of STRs that generate (pseudo-)exhaustive or pseudo-random test patterns for the block under test (BUT) and one STR that is configured as a multiple input signature register (MISR) to evaluate the test responses, when the test unit is processed. If the obtained signature differs from the correct signature, the block is faulty. However, even false test responses may result in the correct signature. This is called error masking or aliasing.

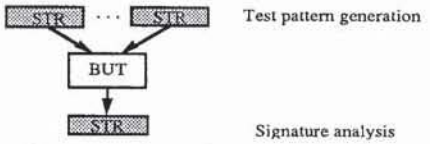

Figure 1: Structure of a test unit

The test of the whole circuit consists of processing all the test units. In this paper it is assumed that the signatures collected in each test unit are retained and the test registers are not reinitialized during the test execution. Then it is sufficient to scan the signatures only at the end of the test, since any difference between the actual contents of an STR and the contents corresponding to the fault-free case will remain unchanged in the pattern generation mode.

A correctly initialized STR can get a faulty signature, if the processed test unit contains a detectable fault, or if at least one of the involved pattern generating STRs has got a faulty signature some time before and thus produces a pattern sequence that differs from the fault-free case. By this mechanism faulty signatures can propagate through the circuit [10].

This can be utilized in two ways. For many circuits the scanning of STRs and the evaluation of signatures can be restricted to a subset of the STRs almost without decreasing the fault coverage. Secondly, test schedules can be constructed that maximize the fault coverage obtained by evaluating only a subset of signatures. This paper presents methods for both issues. The result is a significant reduction of the amount of hardware required to implement a self-test.

Section 2 presents a model that can be used as a basis for the determination of the fault coverage, the selection of signatures, and the test scheduling procedures. Section 3 shows a method to select a minimal number of signatures such that for a fixed test schedule a given fault coverage value is obtained. The problem of constructing a test schedule that gives maximum fault coverage is addressed in section 4, and an algorithm for its solution is presented. Section 5 gives examples and results for some benchmark circuits. A summary and a short discussion in section 6 conclude the paper.

2. Model for fault coverage determination, signature selection, and test scheduling procedures

There is a one-to-one correspondence between the test units and the STRs that can be used as signature registers. Each test unit u_i comprises exactly one STR T_i that is configured as a signature register, and each STR T_i, that can be used as a signature register, is used in exactly one test unit u_i for signature analysis. This correspondence is indicated by the same indices.

The effects of a fault can only propagate in the direction of the data flow, and thus the propagation depends on the circuit structure. The circuit is modeled by the *test register graph* $G_T := (T, E_T)$. Each node $T_i \in T$ represents an STR. The test register graph contains an edge $(T_g, T_i) \in E_T$ for every STR T_g that generates patterns in test unit u_i and thus influences the signature register T_i. The test register graph is independent from the test schedule. Its paths describe how the effects of a fault, namely faulty signatures, can propagate *(propagation paths)*, provided an appropriate test schedule is executed. In the following we explain all the concepts of this

paper with the help of the simple example circuit of figure 2. The test register graph of this circuit is shown in figure 3.

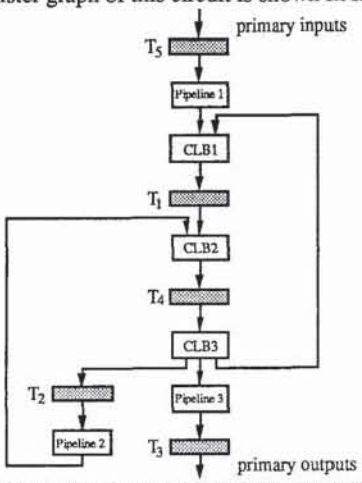

Figure 2: Circuit for matrix multiplication with built-in self-test registers T_i

When the circuit is tested, some test units can be processed in parallel. These test units are called *compatible*. On the other hand, two test units are *incompatible*, if their test hardware requirements are contradictory. For example to ensure that the generated patterns do not depend on the circuit function and are really pseudo-random or pseudo-exhaustive, the contents of an STR that is used as an MISR in one test unit must not be used at the same time as test patterns for another test unit. These and other restrictions due to the limited test resources are modeled by the *test incompatibility graph* $G_I := (U, E_I)$ [2]. The nodes $u_i \in U$ of this graph represent the test units. The test incompatibility graph contains an edge $(u_i, u_j) \in E_I$ if and only if the test units u_i and u_j are incompatible. Figure 3 shows the test incompatibility graph for the example circuit.

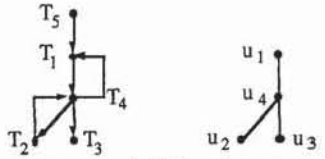

Figure 3: Test register graph (left) and test incompatibility graph (right) for the circuit of figure 2

A test schedule S is represented by a sequence of test sessions $S := (S_0, S_1, ..., S_{d-1})$, where a *test session* S_i is a subset of pairwise compatible test units which are processed in parallel. The test unit with the largest test length determines the test length of the test session. A test schedule for the circuit of figure 2 is $(\{u_1, u_2, u_3\}, \{u_4\})$. The authors of [2] call this the "non-partitioned testing" method. Compared with other, more flexible test scheduling methods discussed by these authors, the test session method needs less hardware for implementing a BIST control unit.

3. Selection of a minimal subset of signatures

In this section the test schedule is fixed and a minimal subset of signatures is determined that gives sufficient fault coverage. Let E(X) be the expectation value of a random variable X, and let F be the set of modeled faults. The *fault coverage FC* is defined by

$$FC := \frac{1}{|F|} \cdot E(\#\text{faults detected by evaluated signatures}).$$

The fault coverage depends on the test schedule and the aliasing probabilities of the signature registers. The aliasing probabilities for single signature registers have been widely investigated, e.g. [3, 4, 7]. Generally in a self-test environment the test lengths for each test unit are long enough to reach the stationary aliasing probabilities of the signature register. In systems with multiple signature registers the method of [10] can be used to compute the expected fault coverage FC, when only a subset of signatures is evaluated at the end of the test execution. When the processing of a test unit is repeated, the resulting signatures are assumed to be statistically independent. This holds if the starting values of the pattern generators are statistically independent.

Of course the fault coverage is maximum, if all signatures are evaluated. But as the signatures influence one another, in most cases a subset is sufficient to obtain almost the same fault coverage.

Problem *Signature Selection*

Given:
- Test register graph G_T
- Set F of faults
- Test schedule S
- Required fault coverage FC_0

Find: A minimal subset of signatures to be evaluated such that $FC \geq FC_0$

The minimal subset must contain all the signatures that cannot influence any other signature. Otherwise all faults that affect only these signatures, would not be covered. Using the information of the test register graph and the test schedule, these signatures can easily be found. In many cases this subset gives sufficient fault coverage. For the remaining cases other signatures are added that are most affected by error masking. A search tree is constructed whose nodes are subsets of the signatures. All nodes at the same level are subsets with the same number of signatures. The subset determined above is the root, the nodes at succeeding levels contain one more signature for each level. When new nodes are added, the successors of the nodes with the highest fault coverage and the smallest subset are prefered. The search stops, when a subset is found that gives the required fault coverage and all the nodes at the previous level have already been examined. Usually only a small portion of the tree must be constructed until a solution is found.

For the circuit of figure 2 with 16-bit-STRs the subset must contain the signature of the STR T_3, because T_3 is located at the primary outputs, and the signature of the STR T_4, since this signature is collected in the last test session. The fault coverage by evaluating these two signatures is less than 10^{-5} below the maximum fault coverage, that would require the evaluation of all the signatures in T_1, T_2, T_3, and T_4.

4. Test scheduling for maximum fault coverage by a minimal subset of signatures

If the test schedule is not fixed, there is an additional degree of freedom to minimize the number of signatures that have to be evaluated. The information in all nonredundant parts of a circuit eventually affects the outputs of the circuit. So all faults can cause faulty signatures in the STRs at the primary outputs provided that the test units are processed in an appro-

priate order. The signatures in the STRs at the primary outputs constitute the minimal subset to be evaluated. But there are circuits where all the propagation paths from some fault locations to the STRs at the primary outputs pass through many other STRs, and in each of them error masking is possible. Thus the fault coverage is adversely affected. The following theorem [11] shows how the problem can be solved.

Theorem:
If the test schedule is a periodically repeated fixed sequence of test sessions, that contain each test unit at least once, the probability that an STR T_i contains a faulty signature after a large number r of repetitions of the test session sequence is

$$\lim_{r\to\infty} P(\text{faulty signature in } T_i) = \begin{cases} 0 & \text{if} \quad T_i \notin T_f \cup s(T_f) \\ 1 - \frac{1}{2^{k_i}} & \text{if} \quad T_i \in T_f \cup s(T_f) \end{cases}$$

where k_i: width of the STR T_i
$T_f \subset T$: set of signature registers in the test units where the fault f is located
$s(T_f)$: set of successors of the nodes of T_f in G_T

The probabilities for a fixed number r of repetitions can be computed exactly as shown in [10]. For long test lengths (many repetitions of the test session sequence) the probability of a faulty signature in each STR T_i approaches a maximum value, that is determined only by the width k_i of the STR. It depends neither on the length of the propagation path from the location of the fault to the STR T_i, nor on the characteristic features of other STRs involved in the propagation process. The results in section 5 will show that the test session sequence must be repeated only few times until the maximum probabilities of faulty signatures are practically reached.

In order to get an inexpensive implementation of the BIST control unit, the test schedule should be composed of a short sequence of test sessions, that is concatenated repeatedly. In this context the scheduling problem can be stated as follows.

Problem *Test Scheduling*

Given:
- Test register graph $G_T = (T, E_T)$
- Test incompatibility graph $G_I = (U, E_I)$
- Set F of faults
- Required fault coverage FC_0

Find: A test schedule $S := (S_0, S_1, ..., S_{rd-1})$
where $S_i \subset U$ and $i \equiv j \bmod d \rightarrow S_i = S_j$
for all $i, j \in \{0, 1, ..., r\cdot d-1\}$
(d test sessions are repeated r times),
such that
1) for all $u_g, u_h \in U$, $S_i \in \{S_0, S_1, ..., S_{rd-1}\}$:
$u_g \in S_i \wedge u_h \in S_i \rightarrow \{u_g, u_h\} \notin E_I$
2) $FC \geq FC_0$
3) d is minimal
4) r is as small as possible for minimal d

Even without the optimization for a small value of r the corresponding decision problem is NP-hard, since it contains the graph K-colorability problem [5] as a special case.

Let O be the set of STRs at the primary outputs. The schedule must guarantee that all (nonredundant) faults can influence the signatures in the STRs of O. Each fault located in a test unit u_{i_1} can cause a faulty signature in the corresponding signature register T_{i_1}. When this STR T_{i_1} is used afterwards to generate patterns for a test unit u_{i_2}, the signature in STR T_{i_2} can also become faulty. The propagation of a faulty signature corresponds to travelling along a propagation path $(T_{i_1}, T_{i_2}, ..., T_{i_s})$ in the test register graph G_T. This propagation is possible only if the corresponding test units $u_{i_1}, u_{i_2}, ..., u_{i_s}$ are processed in the same order. The test schedule must contain a test session comprising the test unit u_{i_1}, then a test session comprising u_{i_2}, and so on. Other test sessions are allowed between these test sessions. Hence the test schedule must look like $(..., \{u_{i_1}, ...\}, ..., \{u_{i_2}, ...\}, ..., ..., \{u_{i_s}, ...\}, ...)$. It can be constructed in two steps:

i) A set of propagation paths is created that contains at least one path from each STR that is used as a signature register to an STR of O.
ii) The test sessions are built such that for every propagation path the corresponding test units appear in the sequence of test sessions in the same order.

Heuristics help to choose an efficient set of propagation paths. From each signature register a *shortest* propagation path to each STR of O is selected. On the shortest propagation path the smallest number of signature registers is involved. As in each of them fault masking is possible, the fault masking probability is often lowest on the shortest path. The shortest paths also contribute to a relatively short overall test length, since in order to propagate a faulty signature along the shortest path the smallest number of test sessions is required. Propagation paths *to all STRs of O that can be reached* are selected, since for all subsets $O'' \subset O' \subset O$ the inequation

P (faulty signature in at least one STR of O'')
$\leq$ P (faulty signature in at least one STR of O') holds.

These heuristics are applied in the algorithm STS ("Self-Test Scheduling", for details see [11]). The inputs are the test register graph G_T, the test incompatibility graph G_I, and the required number d_{min} of test sessions. The algorithm STS first tries to construct a sequence of $d=d_{min}$ test sessions. This is impossible, if d_{min} is less than the chromatic number $\chi(G_I)$ of the test incompatibility graph G_I. If the algorithm does not find such a sequence, the parameter d is incremented until the construction is successful. The output is the test session sequence $S: = (S_0, S_1, ..., S_{d-1})$. If the constructed sequence S does not give sufficient fault coverage, the fault masking probabilities along the propagation paths are reduced by repeating the sequence. The required number r of repetitions can be determined by computing the fault coverage values for some small numbers of repetitions. The complete test schedule is $(S_0, S_1, ..., S_{d-1}, ..., S_{rd-d}, ..., S_{rd-1})$, where $S_i = S_{d+i} = ... = S_{(r-1)d+i}$ for $i = 0, 1, ..., d-1$.

The algorithm STS is demonstrated using the circuit of figure 2 and the corresponding test register graph G_T of figure 3. Only the STR T_3 is scanned, $O = \{T_3\}$. The set of shortest paths in G_T from a signature register to the STR of O is $M := \{(T_1, T_4, T_3), (T_2, T_4, T_3), (T_4, T_3), (T_3)\}$. The STR T_5 is not considered, since it is not used for signature analysis. All propagation paths that are contained in other paths of M can be removed, $M := \{(T_1, T_4, T_3), (T_2, T_4, T_3)\}$.

For $d_{min} = 2$ the constructed test session sequence is $(\{u_4\}, \{u_1, u_2, u_3\})$ with d=2. This is a sequence of minimal length, since $\chi(G_I) = 2$. The sequence must be repeated in order to make the propagation from all signature registers to the STR T_3 possible. The final test schedule is $(\{u_4\}, \{u_1, u_2, u_3\}, \{u_4\}, \{u_1, u_2, u_3\})$.

5. Results

In this section the results of the algorithm STS are compared with the results of the "nonpartitioned testing" algorithm (NT) of [2], that aims at a minimal overall test length and assumes that all signatures are evaluated. The first example E1 is the circuit of figure 2 with 20-bit-STRs, the second example E2 is the same structure but with 8-bit-STRs. In table 1 the results for both scheduling approaches are listed. The number of repetitions is chosen such that in all cases the fault coverage is the same for both scheduling methods. For E1 the error masking probabilities are very low and can be neglected, when the algorithm STS is applied. At the end of the test execution the probability of a faulty signature in the STR T_3 differs from the maximum possible value by less than $2 \cdot 10^{-6}$. This means, the probability that a faulty signature is masked during the propagation to T_3 is less than $2 \cdot 10^{-6}$. If the STRs are 8 bit wide (E2), the probability of a faulty signature decreases along the propagation paths by an amount that cannot be neglected. This is compensated by repeating the test session sequence once more.

circuit	test schedules		#test sessions		total length of signatures (bit)	
	NT	STS	NT	STS	NT	STS
E1	$(\{u_1, u_2, u_3\}, \{u_4\})$	$(\{u_4\}, \{u_1, u_2, u_3\}, \{u_4\}, \{u_1, u_2, u_3\})$	2	2x2	80	20
E2	$(\{u_1, u_2, u_3\}, \{u_4\})$	$(\{u_4\}, \{u_1, u_2, u_3\}, \{u_4\}, \{u_1, u_2, u_3\}, \{u_4\}, \{u_1, u_2, u_3\})$	2	3x2	32	8

Table 1: Test schedules for the example circuits

In the schedules constructed by the algorithm STS, the test lengths are increased by a factor of 2 and 3, respectively, compared to the "nonpartitioned testing" schedule. But the advantage is that for E1 and E2 only the signature in the STR T_3 must be evaluated at the end of the test, whereas with the "nonpartitioned testing" schedule all the signatures of T_1, T_2, T_3, and T_4 must be scanned and evaluated. The number of bits of the signatures derived by STS is significantly less without any loss of fault coverage. There is no need for an internal scan path, as the STR T_3 at the primary outputs can be accessed via the boundary scan chain [6]. The BIST control unit does not have to scan the signatures in the internal STRs and can be simplified. The control unit of the boundary-scan architecture does not need any instructions to control an internal test data register. The comparison of signatures requires less effort. Altogether the amount of test hardware is reduced significantly.

Some large ISCAS'89 benchmark circuits [1] were also investigated. At all primary inputs and primary outputs, boundary scan cells were added. Then STRs were built in, such that all global feedback loops of the circuits were cut by at least two STRs. In some cases this required additional flipflops. Table 2 compares the test scheduling results.

circuit	#gates	#STRs	#test sessions		total length of signatures (bit)	
			NT	STS	NT	STS
s5378	2779	8	7	10	143	49
s9234	5597	4	4	6	335	22
s13207	7951	5	5	6	821	121
s15850	9772	3	3	5	979	87
s35932	16065	5	5	6	1017	320
s38417	22179	4	4	7	2262	106
s38584	19253	6	6	9	2567	278

Table 2: Results for ISCAS'89 benchmark circuits

6. Conclusions

Two methods were presented that reduce the number of signatures that have to be evaluated at the end of the test to a minimum. The first method assumes that the test schedule is given, the second includes the test scheduling process in the optimization. Both of them do not adversely affect the fault coverage compared to the conventional approach of evaluating all signatures.

Generally the schedules obtained by the second method are longer than schedules, that aim at a minimal test execution time. But the advantages are the many hardware savings when a built-in self-test is implemented. If some internal STRs are very small (e.g. less than 6 bit), then propagation paths that contain these STRs have to be repeated many times. This can lead to a long overall test time. Very small STRs at the primary outputs can also cause problems to get a sufficient fault coverage. But in most circuits the STRs are sufficiently wide and do not cause any trouble.

References

[1] F. Brglez, D. Bryan, K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits", Proc. International Symposium on Circuits and Systems ISCAS'89, Portland, Oregon, pp 1929-1934, 1989

[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. 37, 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] W. Daehn, T. W. Williams, K. D. Wagner, "Aliasing errors in linear automata used as multiple-input signature analyzers", IBM Journal of Research and Development, Vol. 34, No. 2/3, March/May 1990, pp 363-380

[5] M. R. Garey, D. S. Johnson, "Computers and Intractability", Freeman, New York 1979

[6] IEEE Standard Test Access Port and Boundary-Scan Architecture, IEEE Std 1149.1-1990, May 21, 1990

[7] A. Ivanov, V. K. Agarwal, "An Iterative Technique for Calculating Aliasing Probability of Linear Feedback Signature Registers", Proc. 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] A. Krasniewski, A. Albicki, "Automatic Design of Exhaustively Self-Testing Chips with BILBO Modules", Proc. International Test Conference ITC-85, pp 362-371, 1985

[10] A. P. Ströle, H.-J. Wunderlich, "Error Masking in Self-Testable Circuits", Proc. International Test Conference ITC-90, Washington 1990, pp 544-552

[11] A. P. Ströle, H.-J. Wunderlich, "Signature Analysis and Test Scheduling for Self-Testable Circuits", Proc. International Symposium on Fault-Tolerant Computing FTCS-21, Montreal 1991

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