BIST SYNTHESIS: AN APPROACH TO RESOURCE OPTIMIZATION UNDER TEST-TIME CONSTRAINTS[1]

Abdil Rashid Mohamed, Zebo Peng and Petru Eles
Embedded Systems Laboratory, Department of Computer Science
Linköping University, S-581 83 Sweden
Emails: {abdmo, zebpe, petel}@ida.liu.se

Abstract. *An approach at optimizing the BIST resource usage under test-time constraints is introduced. The test-problem identification and BIST enhancement strategy during the optimization process are assisted by symbolic testability analysis. Further, concurrent test sessions are generated, while MISRs sharing conflicts as well as controllability and observability constraints are considered.*

1 Introduction

As Automatic Test Equipment (ATE) is slow and expensive, *built-in self-test (BIST)* is the most suitable and cheap method for at-speed testing of complex systems on silicon. A lot of work on BIST at high level has been done [1], [2], [3], [4], [5], [6], [7], [9], [10] and [11], but BIST test time minimization at high level is insufficiently explored. An approach to minimize testing time in a combined BIST and ATE environment was presented [10]. However, the issue of sharing BIST circuitry among cores or functional modules was not studied. The work did not explore parallelism inside the cores to reduce test time during high level synthesis either. An efficient approach for BIST hardware insertion with short test application time is proposed in [11]. It achieves concurrent testing of modules by sharing pattern generators. Both short test application time and low BIST overhead are achieved, but BIST insertion is performed without testability analysis and loss of randomness of test data may happen when some modules are deep in the design.

An ILP formulation for making simultaneous trade-off between test time and BIST resource optimization is proposed in [12]. The approach results in very high BIST hardware overhead and test time minimization is neither sufficiently discussed nor supported by experimental results.

Chen [13] proposed an approach for concurrent test scheduling in a BIST environment. First, he assigned BIST registers to each CUT and then efficiently solved the test-scheduling problem to minimize test time and improve BIST register utilization. BIST register selection is performed without testability analysis; hence no optimal procedure for selecting BIST registers is given. Furthermore, selection of BIST registers and test scheduling are independently performed. [14] Introduced an approach to find an optimal register assignment for testing a design in a given number of test sessions.

Symbolic testability analysis (STA) [2] for BIST leads to very high fault coverage and low hardware overhead. However, the trade-off between hardware overhead and test time still remains to be studied. Most of the current approaches to BIST test time minimization are based on test scheduling optimizations, but efficient test time minimization by sharing BIST components is not well addressed.

Our research on BIST synthesis at high level proposes a systematic approach for designing self-testable SoCs by considering testability issues early in the design process. In this way, functionality and testing can be optimized simultaneously at high abstraction level to reduce design costs and time dedicated to testing effort. Our methodology analyzes the testability of the designs so that hard to test parts can be isolated early during the design process and be optimized for testing.

We use STA to guide BIST synthesis and BIST hardware optimization under testing time constraints. The contribution of this work is twofold. Firstly, STA results are used to guide BIST synthesis and determine which operations can be tested concurrently. Secondly, design modifications are proposed to optimize BIST hardware usage under testing time constraints.

The rest of the paper is organized as follows. In Section 2, the BIST optimization problem is formulated. Section 3 describes STA idea. Our proposed methodology is described in section 4. Section 5 describes our approach to BIST synthesis and resource optimization. In section 6 experimental results are presented and conclusions are drawn in Section 7.

2 Problem Formulation

Initially, all primary input registers are converted into Pseudo-random Pattern Generators (PRPG) and all primary output registers to Multiple Input Signature Registers (MISR). More test registers for BIST enhancement can be used, if necessary.

[1] The work is sponsored by the Swedish Foundation for Strategic Research under the INTELECT program.

The problem is to optimize BIST resources usage under self-test time constraints. The aim is to create a tool to analyze the testability of the design and to determine the minimal possible testing time, T_{min}, which can be achieved as a result of the parallelism inherited by the nature of the design itself. Given a certain required maximum testing time, T_{req}, the following alternatives are taken:

- If $T_{req} < T_{min}$, return no solution;
- If $T_{req}=T_{min}$, optimize BIST hardware, so that minimal overhead is left and return the current testing time, T_{min}, and the modified RTL design;
- If $T_{req}>T_{min}$, optimize the BIST hardware, such that minimal overhead is left and testing time is $T_{BIST}\leq T_{req}$ (T_{BIST} approaches T_{req}), and return T_{BIST} and the modified RTL design.

In summary, the input to our BIST time analysis and resource optimization tool is an RTL design represented in a high-level synthesis notation based on Control Dataflow Graphs. The outputs are: a test schedule shorter than or equal to the test time constraint, an RTL design with minimal added BIST resources and a merged design and BIST controller.

3 Introduction to Symbolic Testability Analysis (STA)

Our approach uses STA to derive control and observation paths for all operations in the design. Since we assume BIST, all justification and propagation paths are computed with respect to the built-in PRPGs and MISRs. Controller and data path are used to extract an intermediate Test Control Data Flow (TCDF) [2] representation that is suitable for deriving a set of symbolic justification and propagation paths, known as Test Environments (TE), for testing a module under test (MUT).

STA defines four Boolean values for controllability and observability of each TCDF variable. General controllability, $C_g(n)$, of a TCDF variable on the n^{th}-control cycle is the ability to control the variable to any arbitrary value from the corresponding PRPGs. Similarly, controllability to the constant value 1, $C_1(n)$, and controllability to the constant value 0, $C_0(n)$, are defined. Observability, $O_v(n)$, of a TCDF variable is the ability to observe any value of the variable at a MISR. If one or several of the controllability values needed to test a module, are *false*, then the associated variable is uncontrollable in the given control step.

The test environments for a MUT are obtained by looking at its input lines and tracing back the propagation paths that can be used to set its values from the primary input ports or PRPGs. To derive the test environments, it is necessary to force intermediate active functional modules to take particular values to assist in propagating test data from PRPGs to MUT and from MUT to appropriate primary output or MISR.

4 Proposed Methodology

Simultaneous analysis of test responses from multiple functional modules requires as many MISRs as there are modules that are to be analyzed at the same time. In addition, all these modules must have all their inputs controlled simultaneously by setting appropriate values on the control variables as given in their TE [2].

Table 1: Alternative TEs for testing *3 and +5

	For controlling operation				***For observing responses***
TEs	**V1**	**V2**	**V3**	**V4**	**V5**
TE1 of op *3	$C_g(1)$	$C_1(1)$	$C_g(1)$	$C_1(1)$	$C_0(1)$
*TE2 of op *3*	$C_g(1)$	$C_1(1)$	$C_1(1)$	$C_g(1)$	$C_0(1)$
TE3 of op *3	$C_1(1)$	$C_g(1)$	$C_g(1)$	$C_1(1)$	$C_0(1)$
TE4 of op *3	$C_1(1)$	$C_g(1)$	$C_1(1)$	$C_g(1)$	$C_0(1)$
TE1of op+5	$C_g(1)$	-	-	$C_g(1)$	-

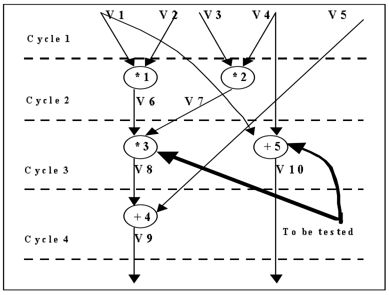

Figure 1: TCDF example.

To illustrate our idea, consider the TCDF in Fig. 1. Inputs and outputs of the operations are variables, and the test environments of each operation are used to test the associated functional module that performs the operation. To test, for example, multiplier node *3 using PRPGs placed at the inputs of operations *1 and *2, and a MISR at the output of +4, we need to control V6 and V7 to general controllability values in the control cycle 2 and observe the value of V8 in control cycle 3. Therefore, test environments for operation *3 are given by *"$C_g(2)_{V6}$ and $C_g(2)_{V7}$ and $O_v(3)_{V8}$"* and are derived as follows. $C_g(2)_{V6}$:={ $C_g(1)_{V1}$ AND $C_1(1)_{V2}$ } OR { $C_1(1)_{V1}$ AND $C_g(1)_{V2}$ }, $C_g(2)_{V7}$:={ $C_g(1)_{V3}$ AND $C_1(1)_{V4}$ } OR { $C_1(1)_{V3}$ AND $C_g(1)_{V4}$} and $O_v(3)_{V8}$:= $O_v(4)_{V9}$AND $C_0(1)_{V5.}$ In total, there are four different alternative test environments for testing *3 (Table 1).

To illustrate the next idea, let us derive TEs for operation +5, see Fig.1, which are given as *"$C_g(2)_{V1}$ and $C_g(2)_{V4}$ and $O_v(3)_{V10}$"*. For left input, $C_g(2)_{V1}$:=$C_g(1)_{V1,}$ for right input, $C_g(2)_{V4}$:=$C_g(1)_{V4}$ and for observability of output, $O_v(3)_{V10}$:=$O_v(4)_{V10}$.

If a given TCDF variable, say V_k, needs to be controlled to the same value in the same control cycle in test environments of different operations, say OP_{FU1}, OP_{FU2}, …, OP_{FUn}, then this common controllability value can be shared by those operations to perform their concurrent testing. For example, consider variables V1 and V4 in the TEs of *3 and +5 as discussed above.

As shown in table 1, both the second test environment alternative of *3 (TE2 of op *3) and the test environment of +5 need V1 and V4 to be controlled to $C_g(1)$. Therefore, V1 and V4 can be shared to perform concurrent testing of both operations using the test environment TE2 of operation *3.

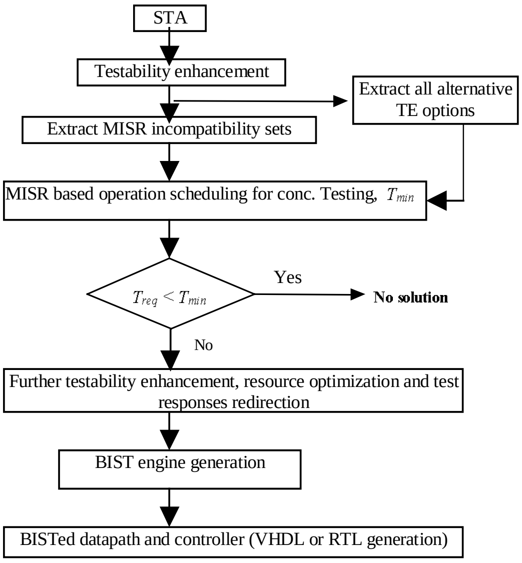
STA
Testability enhancement
Extract all alternative TE options
Extract MISR incompatibility sets
MISR based operation scheduling for conc. Testing, T_{min}
$T_{req} < T_{min}$
Yes
No solution
No
Further testability enhancement, resource optimization and test responses redirection
BIST engine generation
BISTed datapath and controller (VHDL or RTL generation)

Figure 2: Overview of the BIST time analysis and resource optimization

Our approach is described in Fig.2. STA is used to select hard to test modules and initial testability enhancement is performed. In the course of STA all possible TEs for each operation are extracted. Then, sets of operations that cannot be tested concurrently due to MISR sharing conflicts are identified. Heuristics are used to select concurrent test sessions based on test environment options and MISR sharing conflicts. A near optimal shortest schedule is generated and its associated time is denoted as T_{min}. After that, the time constraints are considered and appropriate steps taken as discussed in section 2. To optimize resource usage, one basic idea is to redirect test responses from some operations to other MISRs than those originally assigned to, if the time constraints are not violated and the MISRs allow. This step is repeated until near optimal MISR usage is obtained. Finally, a merged design and BIST controller, and a BIST-ed data path are generated.

5 BIST Synthesis

5.1 Initial Testability Enhancement

STA results show that among the uncontrollable nodes, there is a tendency of certain uncontrollable nodes to induce controllability problems to all successor nodes. Our controllability enhancement strategy, thus, first enhances the node that is the source of controllability problems. Consequently, enhancing one node can improve controllability of most of the successor nodes. We do this by multiplexing the uncontrollable node with a node that is directly controllable from a primary input register or by adding a new dedicated PRPG and multiplex it with the uncontrollable node.

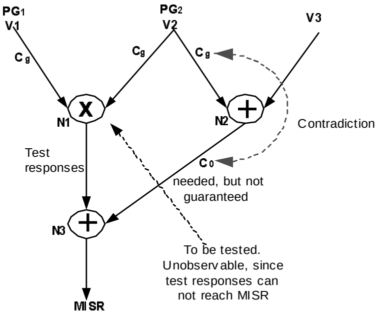
PG1
V1
PG2
V2
V3
C_g
C_g
C_g
N1
N2
Contradiction
Test responses
C_0
needed, but not guaranteed
N3
To be tested. Unobservable, since test responses can not reach MISR
MISR

Figure 3: Observability problem due to contradictory values on intermediate nodes

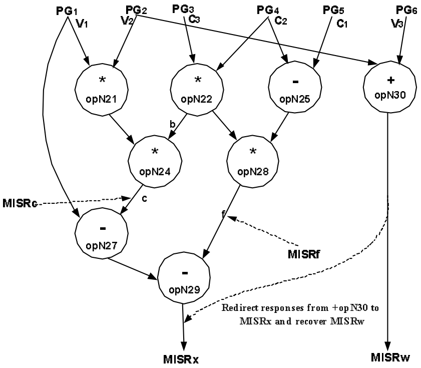

Figure 4: Enhancing observability of unobservable chains and MISR recovery strategy

Unobservable modules are usually buried far from Signature Analysis Registers (SAR). Observability of a node imposes restrictions on the values of other nodes in order for the test responses to be propagated to SARs. Sometimes the restrictions are not able to force propagation of the values to SAR and in some cases some nodes are forced to have contradictory values simultaneously to enable observability, thus, these operations become unobservable as shown in Fig. 3.

If node N1 in Fig. 3 is to be tested, controllability value C_g is to be set at V_1 and V_2 while the output of N2 has to be controlled to C_0 to enable observability of the output of N1 at a MISR. Since V_2 is also connected to N2, whatever value is set at V_3, C_0 can not be guaranteed at the output of N2, hence, test responses at the output of N1 can not reach the MISR.

One solution to the observability problem discussed above is to introduce a MISR at the output of the node N1 or redirect test responses from N1 to an existing MISR in the design. However, in more complex designs, this has to be done in a way such that MISR resources are efficiently used. Therefore, our BIST observability enhancement strategy is to add a dedicated MISR at the output of a node situated at the end of a chain of unobservable nodes. If a MISR is added to an unobservable node that is not at the end of the unobservable chain, then the downstream modules will still be unobservable. This idea is illustrated in Fig. 4. Before BIST enhancement, the design has three primary input variables (V_1, V_2 and V_3) and three constant nodes (C$_1$, C_2 and C_3). STA reveals existence of two unobservable chains. The first one consists of nodes *opN21, *opN24 and *opN22 whereas the second consists of *opN22, *opN28 and –opN25. To enhance the observability of these chains, our approach selects to enhance observability of lines *c* and *f*, which are at the end of the first and second unobservable chains respectively. As a result, observability of all three nodes in each of the two chains is enhanced. Had we, for example, enhanced observability of lines *b* instead, only observability of node *opN22 would have been enhanced. Consequently, it would have been necessary to add more MISRs to improve the observability of the remaining four nodes. Therefore, our approach selects places to enhance observability such that the smallest number of MISRs is added into the design.

5.2 Alternative Test Environment Options

STA reveals the existence of possibly more than one TE for controlling input operands and observing test responses for each operation.

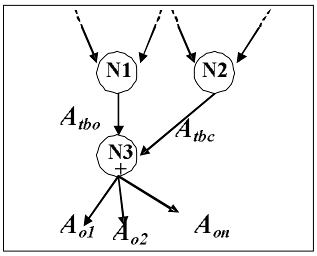

Figure 5: Multiple alternative observability paths

If we want to observe node N1 in Fig.5, we need to observe arc A_{tbo} (A_{tbo} and A_{tbc} stand for arc to be observed and arc to be constrained to controllability value C_0 respectively). Based on STA, this implies constraining A_{tbc} to C_0, and observing the value of A_{tbo} at any of the observable output arc (A_{o1}, A_{o2},…, A_{on}) at the output of node N3. Therefore, the number of observability alternatives increases when the node N3 has multiple observability paths, which, in this case, are also inherited by the node N1, provided that A_{tbc} can be constrained to C_0.

Definitions:

$_{alt}\mathbf{C_0(A_i)}$ is defined as the number of compatible alternative test environment options (ATEO) that can be used to set arc A_i to a controllability value C_0. Similarly, we define $_{alt}\mathbf{C_1(A_i)}$, and $_{alt}\mathbf{C_g(A_i)}$.

$_{alt}\mathbf{O(A_i)}$ is defined as the number of compatible alternative test environment options that can be used to enable observability of an arc A_i at some signature registers.

Two TE alternatives are compatible if and only if each of the Test Control Data Flow (TCDF) variables that are included in both of them needs to be controlled to the same value and at the same control step. However, two ATEOs need not necessarily have exactly the same number and type of variables. They can have some different variables, but the common ones have to be consistent. Therefore, the total number of observability alternative options for arc A_{tbo} can be derived as follows,

$$_{alt}O(A_{tbo}) = {}_{alt}C_0(A_{tbc}) \times {}_{alt}O(A_{o1}) + {}_{alt}C_0(A_{tbc}) \times {}_{alt}O(A_{o2}) + \ldots + {}_{alt}C_0(A_{tbc}) \times {}_{alt}O(A_{on}) \ \ldots \ (1)$$

$$= {}_{alt}C_0(A_{tbc}) \times \sum_{i=1}^{n} {}_{alt}O(A_{oi})$$

Out of these alternatives, that particular TE alternative option which minimizes MISR conflicts and can lead to packing as many operations as possible in each test session will be chosen. Consequently, the total number of test sessions will be minimized. In addition, TEs of all operations in a test session must be simultaneously supported. When the best choice of TE alternatives that give the smallest feasible number of test sessions is achieved, the associated testing time is known as minimal testing time, T_{min}. The best choice among alternative TE options is the one targeted to favor maximum parallelism in testing operations. Section 5.4 discuses a heuristic enhanced with constraints on alternative TE options for concurrent test set selection.

5.3 MISR Incompatibility Sets (MISRISs)

MISR incompatibility sets consist of operations that cannot be tested concurrently due to MISR sharing conflicts. Two operations are contained in the same set if they share the same MISR for test response analysis and, therefore, cannot be tested concurrently. STA results give sufficient information for extracting MISRISs.

To extract MISRISs we group operations based on the signature registers that are used to analyze their responses. Each signature analysis register, M_i, corresponds to one set, G_i, which will include all operations that are analyzed by it. All operations in the same set are known as incompatible operations with respect to their corresponding MISR.

The number of incompatible operations in the largest MISRIS determines a lower bound on the minimal number of test sessions that are needed for testing the whole design. In reality, the total testing time is not only determined by MISR sharing incompatibilities, but also is constrained by the choice of good TE options, which determine whether the TEs are conflict free.

5.4 Concurrent Test Session Selection

Once the MISRISs are available, the next step is to select concurrent test sessions. A group consisting of one operation from each MISRIS can possibly be tested concurrently if the operations will not violate the test environment constraints.

If the test environment constraints are not considered, it can be possible to schedule a minimal number of test sessions equal to the maximum number of operations in the most congested MISRIS. However, these may not be correct test sessions because the availability of MISRs for concurrent observation of responses does not guarantee that those operations can be properly controlled and the responses properly propagated to the corresponding MISR registers at the same time for all tested operations in a given test session. In this way, controllability constraints imposed by the test environments of individual operations may cause an increased number of test sessions. This is due to the fact that there may exist operations that use different MISRs for signature analysis, but compete for the same variables to control their inputs or propagate test response to the appropriate MISR, hence cannot be simultaneously controlled.

Test environments have two components. The first component consists of the controllability values necessary to control the inputs of the operations and the second component consists of the controllability values necessary to force propagation of test responses to the corresponding MISR. Thus, when constraints due to both controllability of the input operands and those imposed to propagate test responses to appropriate MISR are taken into account during the test session selection process, an increase in the number of test sessions will be noticed and the MISRs will be less effectively used, with some of them remaining idle during several test sessions. After all constraints are taken into consideration, the resulting number of test sessions represents the minimal testing time, T_{min}. Thus, it is possible to test the design in T_{min} test sessions as a result of the nature of parallelism inherited from the design itself.

Our heuristic for selection of concurrent test sessions is based on an equal length test-scheduling algorithm in [6], which provides a minimal number of test sessions. We extended the algorithm to take into consideration controllability and observability constraints when choosing operations to be included in a given test session. Therefore, operations are included in the same concurrent test session not only if they do not share MISR, but also if controllability and observability constraints are satisfied for all of them at the same time.

5.5 BIST Resources Optimization

As it has been emphasized in the previous discussion, several MISRs are not effectively used in some test sessions; hence, our approach recovers some of them and converts them back to normal registers. The operations that use recovered MISRs are redirected to other free MISRs in the same test session.

Let L_u represents a MISR that is least used in all test sessions. This means that L_u remains idle in most of the test sessions as compared to other MISRs. Let U be a set consisting of test sessions in which L_u is used. During execution of the algorithm, M_c is the set of currently used MISRs. When a MISR is recovered and converted back to a normal register, it is removed from M_c. F is a set consisting of MISRs that are free in every test session in which L_u is used. Among the free MISRs in set F, P is the one that is mostly packed, which means, P analyzes responses from the greatest number of operations as compared to the other MISRs in F. Let G be the set of all MISR incompatibility sets. Given a certain MISR called X, G_X represents the incompatibility set corresponding to MISR X.

The algorithm below minimizes the set M_C of used MISRs and produces the corresponding incompatibility sets. This optimization is performed without increasing the number of test sessions.

```
Begin
G ← set of all incompatibility sets; Best_selection_obtained ← FALSE;
While (best_selection_obtained != TRUE) begin
    Lu ← X, X ∈ $M_C$ and X is least used; U ← All test sessions in which $L_u$ is used;  F ← Free MISRs in sessions U;
    If F ≠ ϕ begin
        P ← X, X ∈ F and X is most packed; G ← G – { $G_P$, $G_{Lu}$}; $G_P$ ← $G_P$ U $G_{Lu}$; G ← G U { $G_P$}; $M_C$ ← $M_C$ - { $L_u$};
    end
    else best_selection_obtained ← TRUE;
  end
return $M_C$, G;
End.
```

When $T_{req}>T_{min}$, our approach increases the testing time from T_{min} to T_{BIST} by stretching the test schedule such that the required time constraints are satisfied ($T_{min}<T_{BIST}\leq T_{req}$). In this case, we can recover more BIST hardware resources that may not necessarily be needed. The success of this depends on test environment conflicts of the operations and on how large T_{req} is compared to T_{min}.

Additional multiplexers and wiring will be needed in order to redirect test responses for analysis by different MISRs. After some MISRs are disabled as BIST registers, they will still remain in the design as normal registers for their functional storage use, hence not adding any BIST overhead.

6 Experimental Results

We manually tested our approach on four high-level benchmarks, which were synthesized by CAMAD [15].

In our results, testability is computed as a percentage of controllable or observable operations/modules. Controllability, therefore, is the ratio of the number of controllable operations to the total number of operations in the design. Similarly, observability is defined as the ratio of the number of observable operations to the total number of operations in the design. In order for an operation to be counted as controllable, both its left and right hand operands must be simultaneously controllable. If any input operand is not fully controllable, the associated operation is assumed to be not controllable.

Table 2 shows results before testability enhancement and optimization, whereas Table 3 shows results after testability enhancement and optimization. In Table 2, the first column shows the names of the designs and the number and type of functional modules in the designs, the second column shows the number of test sessions required to test the design, and the third column summarizes the number of PGs, MISRs and testability of the design as proposed after the original application of STA, but before our testability enhancement is applied. Testability is depicted in two separate sub-columns. The first one depicts the percentage of modules/operations that are fully controllable and the second sub-column gives the percentage of operations that are observable. The total number of PGs and MISRs after enhancement to 100% testability is shown in table 3 whose second column depicts the number of PGs and MISRs after initial straightforward testability enhancement is performed and the third column gives the number of PGs and MISRs that remain in the design after our BIST resource optimization and MISR recovery strategy is applied. In all our experimental results we have considered that $T_{req} = T_{min}$.

Table 2. BIST resources after applying STA, but before testability enhancement and optimization

Design		*Test sessions (T_{min})*	*Applying original STA*			
					%Testability	
Name	*Operations*		*#PG*	*#MISR*	*Con.*	*Obs.*
Ex	1+, 3-, 4*	5	3	2	12.5	37.5
Tseng	4+, 2*, 1 /, 1 &	4	5	3	100	37.5
Pauln	2+, 2-, 6*	6	4	3	50	50
Diff	2+, 2-, 6*	7	4	3	40	70

Table 3. BIST resources after testability enhancement and optimization to 100% testability

Design	*100% Testability enhancement approaches*			
	Straightforward		*Optimized*	
Name	*#PGs*	*#MISRs*	*#PGs*	*#MISRs*
Ex	6	4	6	3
Tseng	5	5	5	3
Paulin	5	5	5	3
Diff	5	4	5	2

7 Conclusion

An approach to use STA to guide BIST synthesis under testing time constraints has been proposed. STA reveals hard to test parts whose testability needs to be enhanced. The testability enhancement technique we use chooses one module to enhance controllability so as to improve controllability of a number of others. Similarly, observability of one module is normally enhanced to improve observability of a number of others. Further, the design is modified such that the use of BIST resources is optimized under the given testing time constraints.

References

[1] *Boubezari, S.; Cerny, E.; Kaminska, B.; Nadeau-Dostie, B., "Testability Analysis and Test-Point Insertion in RTL VHDL Specifications for Scan-Based BIST", IEEE Trans. on CAD of ICs and Systems, Vol. 18 9, Sept. 1999, Page(s): 1327 –1340.*

[2] *Ghosh,I.; Jha, N.K.; Bhawmik,S, "A BIST Scheme for RTL Circuits Based on Symbolic Testability Analysis", IEEE Transactions on CAD of Integrated Circuits and Systems, Volume: 19 Issue: 1 , Jan. 2000 Page(s): 111 –128.*

[3] *Ravi, S.; Jha, N.K.; Lakshminarayana, G., "TAO-BIST: A Framework for Testability Analysis and Optimisation of RTL Circuits for BIST", In Proc. 17th IEEE VLSI Test Symposium., 1999 , Pages: 398 –406.*

[4] *Ghosh, I.; Jha, N.; Bhawmik, S., " A BIST Scheme for RTL Controller-Data Paths Based on Symbolic Testability Analysis", DAC, 1998 Page(s): 554 –559.*

[5] *Ravi,S.; Lakshminarayana,G.; Jha, N.K.," TAO: Regular Expression based High-Level Testability Analysis and Optimization", In Proc., ITC, 1998, Page(s): 331 –340.*

[6] *Gary L. Craig, C. R. Kime and K. Saluja, "Test Scheduling and Control for VLSI Built-In Self-Test", IEEE Tr. on Computers, Volume: 37 Issue: 9 , Sept. 1988 Page(s): 1099 –1109.*

[7] *Vahidi, M.; Orailoglu, A, "Testability Metrics for Synthesis of Self Testable Designs and Effective Test Plans", In Proc., VLSI Test Symposium, 13th IEEE, 95, Page(s): 170 –175.*

[8] *Vahidi, M.; Orailoglu, A., "Metric-Based Transformations for Self-testable VLSI Designs with High test Concurrency", In Proc. EURO-DAC '95., 1995 , Page(s): 136-141.*

[9] *Harris, I.G.; Orailoglu, A., "Fine-Grained Concurrency in Test Scheduling for Partial-Intrusion BIST", EDATC 1994 , P. 119-123.*

[10] *Sugihara, M.; Date, H.; Yasuura, H., "Analysis and Minimization of Test Time in a combined BIST and External Test Approach", In Proc., Design, Automation and Test in Europe 2000, Pages: 134 –140.*

[11] *Nicolici, N.; Al-Hashimi, B.M., "Efficient BIST Hardware Insertion with Low Test Application Time for Synthesized Data Paths", In Proc., Design, Automation and Test In Europe Conference and Exhibition 1999, page(s) 289-295.*

[12] *Li, X.; Cheung, P.Y.S., "Exploiting Test Resource Optimization in Data Path Synthesis for BIST", In Proc., 9th Great Lakes Symposium on VLSI, 1999 Page(s): 342 –343.*

[13] *Chen, C. H.; Yuen, J.T., ,,Concurrent test Scheduling in Built-In Self-Test Environment", In Proc. ICCD'92 VLSI in Computers and Processors, 1992, Page(s): 256 –259.*

[14] *Kim, H. B.; Takahashi, T.; Ha, D. S., "Test Session Built-In Self-testable Data Path Synthesis", ITC, 1998, Page(s): 154 –163.*

[15] *Peng,Z.; Kuchcinski,K., "Automated Transformation of Algorithms into Register-Transfer Level Implementations", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 13, 150-166, 1994.*