

Deterministic BIST with Partial Scan

Gundolf Kiefer Hans-Joachim Wunderlich

Computer Architecture Lab

University of Stuttgart

E-Mail: Gundolf.Kiefer@informatik.uni-stuttgart.de, wu@informatik.uni-stuttgart.de

Abstract

An efficient deterministic BIST scheme based on partial scan chains together with a scan selection algorithm tailored for BIST is presented. The algorithm determines a minimum number of flipflops to be scannable so that the remaining circuit has a pipeline-like structure. Experiments show that scanning less flipflops may even decrease the hardware overhead for the on-chip pattern generator besides the classical advantages of partial scan such as less impact on the system performance and less hardware overhead.

1. Introduction

The attractiveness of a Built-In Self-Test (BIST) scheme is mainly determined by the achievable fault coverage and the impact of the test hardware on the system performance of the circuit under test. Furthermore, the hardware overhead and the test application time should be within an acceptable range.

Complete fault coverage can be achieved by implementing an on-chip pattern generator for deterministic patterns [1-5]. To minimize the impact on the system performance, a test-per-scan scheme [6, 7] is often preferred compared to a test-per-clock scheme [8].

However, even a scan design requires additional hardware and may reduce the system performance if the scan elements are located in the critical path. If a partial scan path [9] is used, these problems can be reduced and more test patterns may be applied within the same test application time as less clock cycles are needed to load the scan chain.

State-of-the-art techniques for selecting partial scan flipflops are based on testability analysis [9-11], test generation [12, 13], or on structural analysis [14-19]. A survey on partial scan selection techniques is given in [20].

The purpose of this work is to evaluate the impact of a partial scan design on the effort required to implement a deterministic BIST with complete fault coverage. A BIST scheme and a scan selection algorithm are described that allow to implement a partial scan BIST at lower hardware costs than a BIST based on full scan.

In section 2, the target structure for the scan-based deterministic BIST scheme is described, and its properties with respect to partial scan are discussed. In sections 3 and 4, a scan selection strategy is derived and an optimal algorithm for finding a set of flipflops to scan is described. Experimental results are presented in section 5.

2. Target Structure

The target structure is shown in figure 1. The pattern generator contains an LFSR generating pseudo-random patterns. A Sequence-Modifying Logic (SML) is used to modify the random sequence at a few bit positions so that deterministic patterns are generated and complete fault coverage can be achieved.

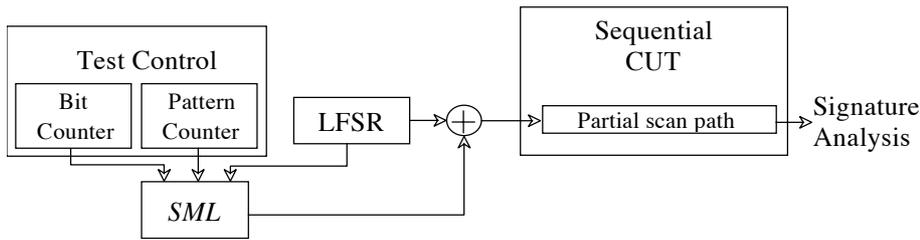


Figure 1: Target Structure

An automatic procedure for synthesizing the SML is described in [3]. As the generated signal depends on the state of the test control unit the LFSR can be very small, and it has been shown that the complete pattern generator including the SML may be smaller than a 32-bit LFSR while achieving complete fault coverage.

Figure 2 sketches how the SML for embedding a single deterministic test is constructed. In the example, the circuit under test contains a partial scan path with 4 flipflops, the test of a certain fault requires a sequence of two patterns <"-0-", "-110">, where "-" are unspecified bits. The states of the LFSR and the test control unit are denoted as S_0, \dots, S_{15} . Each possible mapping of the deterministic pattern to a pseudo-random pattern is characterized by two state sets *ON* and *OFF* corresponding to bits that have to be modified and bits that must not be modified. These sets serve as an on-set and an off-set and specify the SML. All states not contained in *ON* or *OFF* are don't cares and can be exploited for logic minimization [21].

State:	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}
from LFSR:	0	1	1	0	1	0	0	0	1	0	1	0	0	0	1	0
a)	-	-	0	-	-	1	1	0	-	-	-	-	-	-	-	-
b)	-	-	0	-	-	-	0	-	-	1	1	0	-	-	-	-
c)	-	-	0	-	-	-	0	-	-	-	1	1	0	-	-	-
														<i>ON</i>	<i>OFF</i>	
														{ S_2, S_5, S_6 }	{ S_7 }	
														{ S_9 }	{ S_6, S_{10}, S_{11} }	
														{ S_{10}, S_{13} }	{ S_{14}, S_{15} }	

Figure 2: Possible embeddings of a deterministic pattern sequence <"-0-", "-110">

The size of the minimized SML depends on the size of the on-set and the off-set which determine the size of the don't-care set. Furthermore, to obtain a small SML, *ON* and *OFF* should be different in size as much as possible. So the efficiency of the BIST scheme depends on two facts:

1. The total number of specified bits in the deterministic pattern. A small number of specified bits leads to small on- and off-sets.
2. The total number of patterns that can be shifted in within a given test application time. A larger number of patterns leads to more alternatives for pattern mapping, and at a high probability an embedding can be found where the on-set is much smaller than the off-set [22].

3. Scan selection aspects for BIST

To allow an efficient BIST implementation, the scan flipflops should be selected such that their number is small and at the same time a small number of specified bits for the test of each fault can be guaranteed.

In [15] it has been observed that the test generation complexity of sequential circuits is mainly due to feedback cycles with more than one flipflop. However, even if the circuit only contains self-loops with one flipflop, the number of specified bits may grow exponentially with the number of flipflops for some faults. For example, this is the case for a synchronous counter based on half adders, if a stuck-at-0 fault at the most significant output has to be tested.

If the circuit under test does not contain any cycles at all, the number of specified bits for any fault is limited by $n \cdot |PPI|$, where n is the maximum number of flipflops on any path and PPI is the set of all pseudo-primary inputs (= primary inputs and scanned flipflops).

The total number of specified bits is limited by $|PPI|$ if the circuit is acyclic and equidistant [16, 19]. A circuit is called *equidistant*, if there are no nodes between which paths with different numbers of flipflops exist, e. g. every pipeline-structured circuit is equidistant. Figure 3a shows an example of an equidistant circuit. All faults can be tested by a sequence of two patterns where line c may be specified in the first pattern (but not in the second) and lines a and b may be specified in the second pattern (but not in the first). The total number of specified bits will never exceed 3. The circuit in figure 3b is not equidistant. It contains a so-called *asymmetric reconvergence* (bold lines) between c and f which causes f to depend on the value of c in two different system clock cycles.

The problem of breaking all cycles optimally is NP-complete [23]. In [17] it is shown that breaking all asymmetric reconvergences optimally in an acyclic graph is NP-hard. In consequence, finding a minimum set of flipflops for breaking all cycles and asymmetric reconvergences is NP-hard, too.

In both [16] and [19], heuristic scan selection algorithms for obtaining acyclic and equidistant circuits have been described. It has been proposed to first cut all cycles and then to resolve asymmetric reconvergences of the remaining acyclic circuit. For the first problem a fast and optimal algorithm is available [14], for the second problem only heuristics are known which find suboptimal solutions within reasonable computation time. However, even if an efficient optimal algorithm would be available, the overall solution would still be suboptimal due to the two-step approach.

In the next section a fast algorithm is described which solves both problems in one step and achieves a provably optimal solution for all ISCAS'89 benchmark circuits [24] within a few seconds of CPU time.

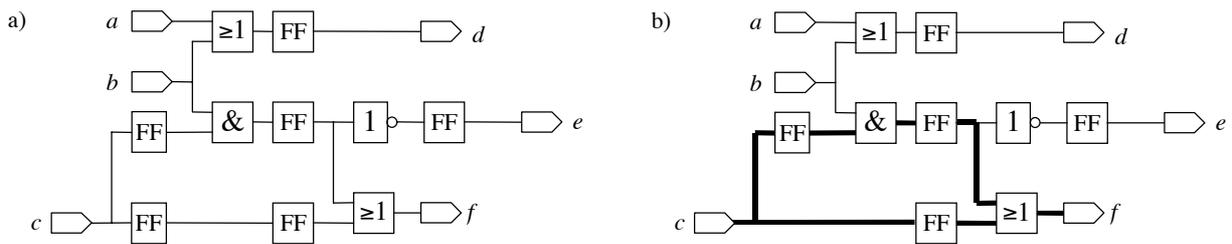


Figure 3: Equidistant & non-equidistant circuit

4. Determining a minimum set of scan flipflops

The *S-graph* of a circuit is a directed graph $G = (V, E)$, where V contains all flipflops, primary inputs and primary outputs, and $(v_1, v_2) \in E$ if and only if there is a combinational path from v_1 to v_2 [15, 19].

A pair of paths $R = (\langle d, v_1, v_2, \dots, v_n, c \rangle, \langle d, w_1, w_2, \dots, w_m, c \rangle)$, where $d, c, v_1, \dots, w_1, \dots, w_m \in V$, is called an *asymmetric reconvergence* if the paths are different in length, acyclic and do not share any nodes except for d and c (where $d = c$ is possible). In the sequel, d is called a *divergence node*, and c is called a *convergence node*.

For example, figure 4 shows an S-graph containing the following asymmetric reconvergences: $R_1 = (\langle b, e \rangle, \langle b, c, e \rangle)$, $R_2 = (\langle a, b, c, e \rangle, \langle a, d, e \rangle)$, $R_3 = (\langle a, b, e, g, d \rangle, \langle a, d \rangle)$ and $R_4 = (\langle a, b, c, e, g, d \rangle, \langle a, d \rangle)$. The set of all divergence nodes is $D = \{a, b\}$, the set of all convergence nodes is $C = \{d, e\}$.

A *minimum pipelining vertex set (MPVS)* of an S-graph $G = (V, E)$ is a set of nodes $S \subseteq V$ so that the remaining circuit is acyclic and does not contain any asymmetric reconvergences if the nodes corresponding to S are scanned. A cycle in the original S-graph can be resolved by putting any of its nodes into S . An asymmetric reconvergence can be resolved by putting any of its nodes except its convergence and its divergence node into S . In the example shown in figure 4, the set $\{c, g\}$ is an MPVS.

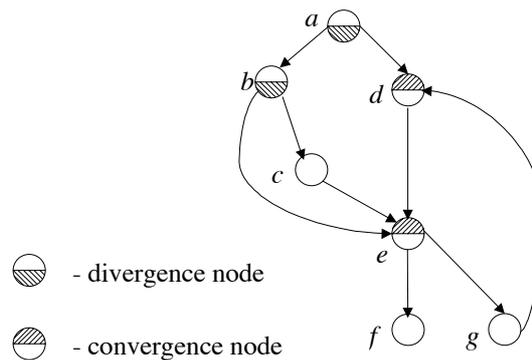


Figure 4: S-graph with asymmetric reconvergences

The algorithm is based on graph reduction, graph partitioning and exhaustive search. It requires the following subsets of nodes representing certain node attributes:

$D \subseteq V$: set of all divergence nodes

$C \subseteq V$: set of all convergence nodes

$Z \subseteq V$: set of "zombie" nodes. These are nodes for which the algorithm has already decided if they belong to the MPVS or not, but which cannot be removed as they are still part of an unresolved asymmetric reconvergence.

In order to keep track of the path lengths of the original circuit after reduction, the edges are labeled with the original distance $d(v_1, v_2)$. Initially $d(v_1, v_2) = 1$ for all $(v_1, v_2) \in E$. Whenever the S-graph is modified (e. g. by graph reduction, see below), the distances are maintained so that $d(v_1, v_2)$ is the number of edges of any path from v_1 to v_2 in the original circuit.

4.1. MPVS-preserving graph reduction and partitioning

A set of rules is established to check which nodes v are not necessary for an optimal solution and which of them are essential for a correct solution. The graph is then reduced using two procedures $Remove(v)$ and $Ignore(v)$ respectively.

$Remove(v)$ is usually performed on nodes known to belong to the MPVS. The procedure deletes v and all incident edges. If v was a divergence (convergence) node, the incoming (outgoing) edges are not deleted, and v is marked as a "zombie". $Remove(v)$ will then be called again each time its convergence or divergence status changes. Figure 5 shows two examples for the effect of $Remove(v)$.

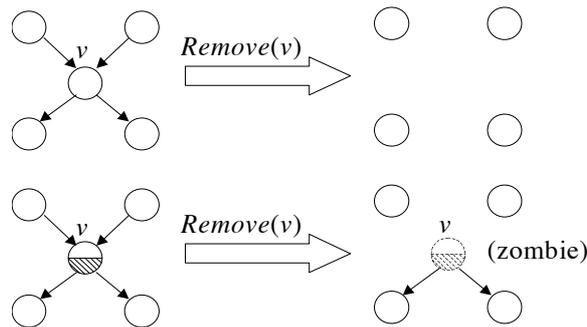


Figure 5: Examples for $Remove(v)$

$Ignore(v)$ is usually performed on nodes which do not belong to the MPVS. The procedure bypasses v by connecting all its predecessors with all its successors and then removes it. The sets D and C are updated, so that if v is a divergence node, all its predecessors become divergence nodes, and if v is a convergence node, all its successors become convergence nodes. Figure 6 shows an example for $Ignore(v)$.

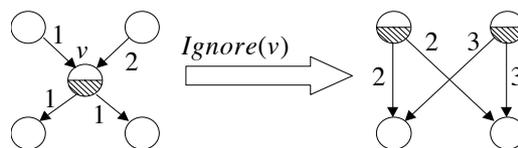


Figure 6: Example for $Ignore(v)$

Let $deg_{in}(v)$ be the number of predecessors and $deg_{out}(v)$ be the number of successors of v . The rules for reducing the S-graph are as follows:

Rule DEG-0a(b): If there is a node v so that $deg_{in}(v) = 0$ ($deg_{out}(v) = 0$) then $Remove(v)$

Rule DEG-1a(b): If there is a node v so that $deg_{in}(v) = 1$ ($deg_{out}(v) = 1$), $(v, v) \notin E$ and its predecessor (successor) is not a divergence (convergence) node then $Ignore(v)$

Rule LOOP: If the graph contains a self-loop $(v, v) \in E$ then put v into the MPVS, delete (v, v) from E and $Remove(v)$

Rule RECO: If the graph contains nodes u, v, w so that $\{(u, w), (u, v), (v, w)\} \subseteq E$ and $d(u, w) \neq d(u, v) + d(v, w)$ then put v into the MPVS and $Remove(v)$

Figure 7 shows an example for the complete reduction of a graph by using these rules. Node a is not deleted in step 1 after applying rule LOOP. Instead, it is turned into a zombie and deleted later in step 2 after the asymmetric reconvergence $\langle a, b \rangle, \langle a, c, b \rangle$ has been resolved.

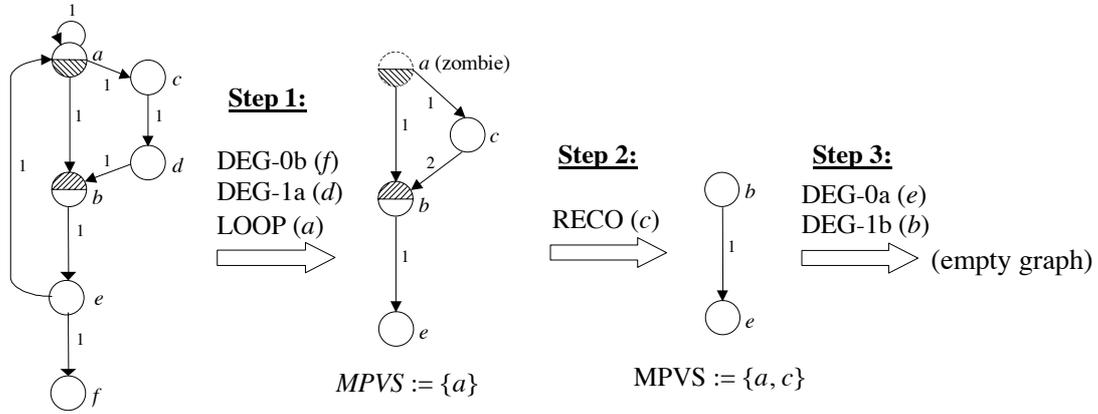


Figure 7: Determining the MPVS by graph reduction

If complete reduction is not possible, the graph is partitioned into components that can be processed independently. First, the strongly connected components (SCCs) are computed. Cycles cannot cross SCC boundaries, but asymmetric reconvergences can do so in some cases. So the following rule is iteratively applied as long as possible. It checks if an asymmetric reconvergence can leave or enter a certain SCC and removes edges if possible.

Rule S: If there is a strongly connected component SCC_i so that there are no edges arriving at (leaving) SCC_i and

- 1) exactly one edge leaves (arrives at) SCC_i or
- 2) SCC_i does not contain any divergence (convergence) nodes

then remove all edges arriving at or leaving SCC_i .

Figure 8 shows an example where the rule can be applied as follows: S1 (e_1) - S1 (e_2) - S2 (e_7, e_8) - S1 (e_6).

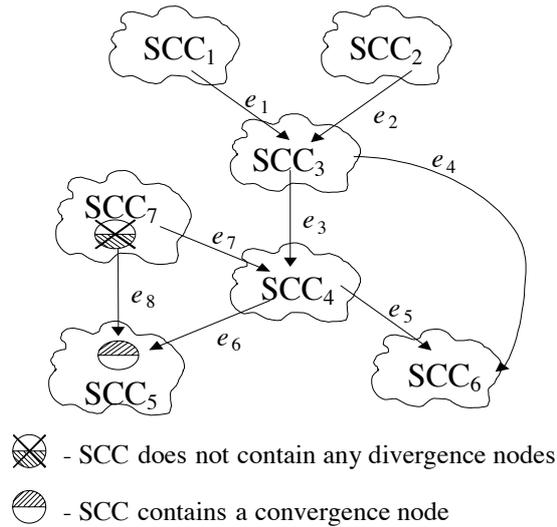


Figure 8: Removing edges between strongly connected components

After removing the edges, some of the simplifying rules can probably be applied again. Then it may be possible to delete some more edges between SCCs and so on. Finally, all weakly connected components are handled separately.

4.2. Branch and bound

If the current S-graph cannot be reduced further, no (non-zombie) node is essential for a correct solution. Now a node v is selected and recursively the best solution containing this node and the best solution not containing it are computed. For computing the best solution containing v , $Remove(v)$ is performed. Then the graph is reduced and partitioned again as described in the previous section. For computing the best solution not containing v , $Ignore(v)$ is used, respectively. In order to save computation time, a global variable $maxCost$ stores the smallest solution found so far, and the search is stopped, whenever the size of the current solution reaches $maxCost$.

Figure 9 shows an example. The initial graph cannot be reduced any further, so branching is performed on node a . After removing / ignoring a the resulting graphs can be reduced completely. The second branch performing $Ignore(a)$ is cancelled after applying RECO on b as the current MPVS has reached a size equal to the best known complete solution $MPVS_1$.

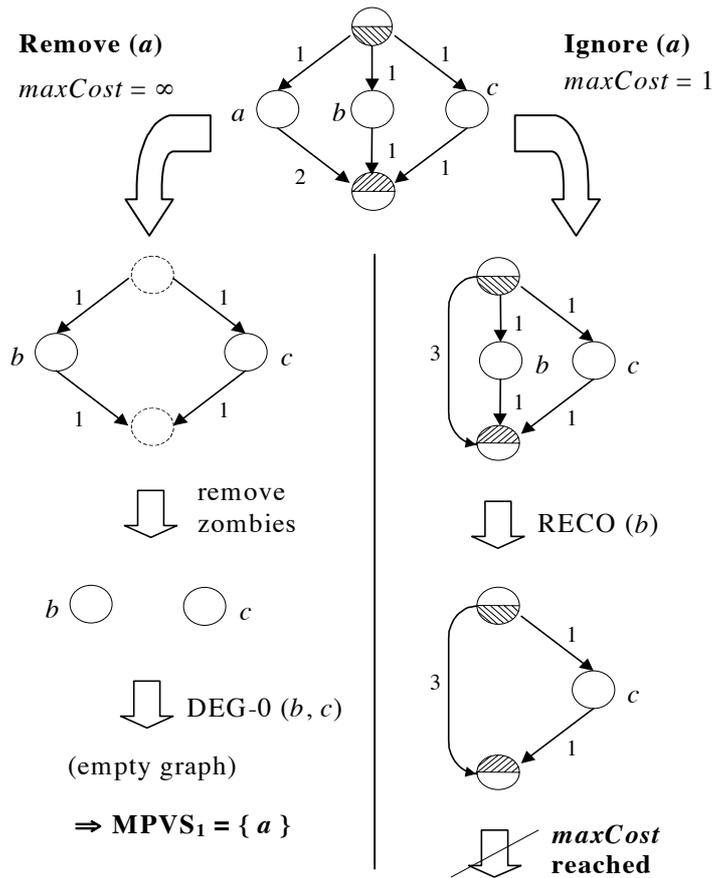


Figure 9: Solving an irreducible S-graph

5. Experimental Results

We applied our scan selection algorithm to all ISCAS'89 benchmark circuits [24]. The results are shown in table 1. The first two columns show the circuit name and the total number of flipflops. The column "Acyclic" is the minimum number of flipflops needed for breaking all cycles using the algorithm described in [14]. The column "MPVS" denotes the size of an optimum set of scan flipflops for breaking all cycles and asymmetric reconvergences. Finally, the required computation time on an UltraSparc 2 is listed in the rightmost column. Circuits requiring a full scan path to make them acyclic are not listed.

For the larger circuits (≥ 5378), the size of the MPVS is typically similar to the average of the "acyclic" number and the total number of flipflops. For some circuits the MPVS numbers are high compared to other scan selection strategies (70% - 90% of all flipflops have to be scanned). However, they refer to provably minimum scan sets so that the total number of specified bits per test is limited by the number of pseudo-primary inputs, and thus an efficient BIST is possible.

Circuit	FFs	Acyclic	MPVS	CPU secs.
s382	21	15 (71%)	15 (71%)	< 0.1
s444	21	15 (71%)	15 (71%)	< 0.1
s641	19	15 (79%)	15 (79%)	< 0.1
s713	19	15 (79%)	15 (79%)	< 0.1
s953	29	6 (21%)	6 (21%)	< 0.1
s1196	18	0 (0%)	16 (89%)	< 0.1
s1238	18	0 (0%)	16 (89%)	< 0.1
s1423	74	71 (96%)	72 (97%)	0.1
s5378	179	30 (17%)	99 (55%)	13.4
s9234	228	152 (67%)	209 (92%)	1.0
s13207	669	310 (46%)	451 (67%)	3.7
s15850	597	441 (74%)	534 (89%)	8.4
s35932	1728	306 (18%)	1728 (100%)	12.6
s38417	1636	1080 (66%)	1207 (74%)	44.6
s38584	1452	1115 (77%)	1433 (99%)	37.1

Table 1: MPVS size of all ISCAS'89 circuits

Using a tool for automatic test pattern generation (ATPG), we investigated the testability of the larger circuits. For the "acyclic" case, a sequential ATPG is used, and pattern generation may become intractable for larger designs. For circuits with an MPVS scan chain, combinational ATPG can easily be applied [16, 19], and efficient algorithms are available.

The columns of table 2 show the circuit name, the total number of collapsed stuck-at faults, the number of untestable faults due to combinational redundancies, and the number of sequentially untestable faults for the "acyclic" and the "MPVS" case. If the remaining circuit is just acyclic, there are always several untestable faults due to sequential redundancies. In the MPVS case almost all combinationally testable faults remain testable. In consequence, better defect coverage may be obtained. The number of sequentially untestable faults is zero if the scan selection strategy is modified in a way that both asymmetric and symmetric reconvergences are broken [18]. Such a set of flipflops can easily be computed using the MPVS algorithm by slightly modifying the definition of convergence and divergence nodes and the simplifying rule RECO.

Circuit	Faults	Comb. untestable	Seq. untestable (Acyclic)	Seq. untestable (MPVS)
s5378	4437	40	239	4
s9234	6927	452	22	0
s13207	9759	151	130	0
s15850	11719	389	3	0
s38417	31024	165	4	0
s38584	35999	1506	179	8

Table 2: Testability of circuits with partial scan chains

Using the algorithm described in [3], we synthesized BIST pattern generators which achieve complete fault coverage of all detectable stuck-at faults. All circuits which still have undetected non-redundant faults after applying 10,000 random patterns in the MPVS or full scan case were investigated.

For each circuit, the test application time is limited by the time required for a full scan test with 10,000 patterns. The chip area of the pattern generator is measured by using a 1 μm standard cell library, and the SML is implemented by PLA macro cells.

The results are shown in table 3. For each case (full scan, MPVS, and "acyclic"), the column "patgen" denotes the area of the pattern generator. For the partial scan cases, the columns "scan savings" describe the area saved by not scanning some flipflops. These numbers are based on the difference in size between a normal D-flipflop cell and a scan-flipflop cell. The wiring is not considered, so in practice, the area savings are usually larger. The columns "total savings" denote the sum of the scan savings and the area saved (or not saved) due to a smaller (or sometimes larger) pattern generator.

Circuit	Full scan	Partial Scan (MPVS)			Partial Scan (acyclic)		
	patgen	patgen	scan savings	total savings	patgen	scan savings	total savings
s641	0.061	0.059	0.004	0.006	(same as MPVS)		
s713	0.061	0.060	0.004	0.005	(same as MPVS)		
s953	0.058	0.056	0.025	0.027	(same as MPVS)		
s1196	0.064	0.062	0.002	0.004	0.073	0.020	0.011
s1238	0.062	0.059	0.002	0.005	0.077	0.020	0.005
s1423	0.060	0.057	0.002	0.005	0.054	0.003	0.009
s5378	0.087	0.082	0.088	0.093	0.177	0.164	0.074
s9234	0.474	0.442	0.021	0.053	0.428	0.084	0.130
s13207	0.152	0.198	0.240	0.194	0.267	0.396	0.281
s15850	0.287	0.300	0.069	0.056	0.743	0.172	-0.284
s38417	1.344	1.553	0.473	0.264	2.381	0.613	-0.424
s38584	0.255	0.254	0.021	0.022	0.484	0.371	0.142

Table 3: BIST area overhead [mm²] for different scan selection strategies

For the MPVS scan selection these numbers are always positive which means that a partial scan BIST is more efficient than a full scan BIST. If the scan flipflops are selected such that all cycles are broken, the results depend on the circuit. In some cases, the savings are even larger than for MPVS scan insertion, but with some larger circuits (e. g. s15850, s38417), the hardware overhead increases drastically.

For each case all detectable faults are considered. Please remember that in the "acyclic" case there are often a large number of sequentially untestable faults. So the "acyclic" numbers may be optimistic: On the one hand, the defect coverage may be lower in this case, on the other hand, the same defect coverage for the "full scan" and "MPVS" case would require a smaller pattern generator than the one stated in the table.

6. Conclusions

A deterministic BIST scheme based on partial scan together with a fast and optimal scan selection algorithm for providing acyclic and equidistant circuits has been described. Experimental results have shown, that for all ISCAS'89 benchmark circuits the area overhead for partial scan BIST is less than for a full scan BIST without any reduction of the fault coverage.

References

- [1] S. Hellebrand, B. Reeb, S. Tarnick, H.-J. Wunderlich: "Pattern Generation for a Deterministic BIST Scheme", Proc. Int. Conf. on Computer-Aided Design (ICCAD), 1995, pp. 88-94
- [2] G. Kiefer, H.-J. Wunderlich: "Deterministic BIST with Multiple Scan Chains", Proc. of International Test Conference (ITC), 1998, pp. 1057-1064
- [3] G. Kiefer, H.-J. Wunderlich: "Using BIST Control for Pattern Generation", Proc. IEEE Int. Test Conf. (ITC), 1997, pp. 347-355
- [4] B. Koenemann: "LFSR-Coded Test Patterns for Scan Design", Proc. Europ. Test Conf. (ETC), Munich 1991, pp. 237-242
- [5] N. A. Touba, E. J. McCluskey: "Altering a pseudo-random bit sequence for scan-based BIST", Proc. Int. Test Conf. (ITC), 1996, pp.167-175
- [6] E. B. Eichelberger, E. Lindbloom: "Random Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Tet", IBM Journal of Research and Development, Vol. 27, No. 3, May 1983, pp. 265-272
- [7] P. H. Bardell, W. H. McAnney: "Parallel Pseudo-random Sequences for Built-In Test", Proc. Int. Test Conf. (ITC), 1984, pp. 302-308
- [8] B. Koenemann, J. Mucha, G. Zwiehoff: "Built-In Logic Block Observation Techniques", Proc. of International Test Conference (ITC), 1979
- [9] E. Trischler: "Incomplete Scan Path with Automatic Test Generation Methodology", Proc. International Test Conference (ITC), 1980, pp. 153-162
- [10] M. Abramovici, J. J. Kulikowski, R. K. Roy: "The Best Flipflops to Scan", Proc. International Test Conference (ITC), 1991, pp. 166-173
- [11] F. Corno, P. Prinetto, M. Sonza Reorda, M. Violante: "Exploiting Symbolic Techniques for Partial Scan Flip Flop Selection", Design, Automation and Test in Europe (DATE), 1998, pp. 670-677

- [12] V. Chickermane, J. H. Patel: "A Fault Oriented Partial Scan Design Approach", Proc. of Intl. Conf. On Comp.-Aided Design (ICCAD), 1991, pp. 400-403
- [13] D. Xiang, J. H. Patel: "A Global Algorithm for the Partial Scan Design Problem Using Circuit State Information", Proc. International Test Conference (ITC), 1996
- [14] S. T. Chakradhar, A. Balakrishnan, V. D. Agrawal: "An Exact Algorithm for Selecting Partial Scan Flipflops", Proc. of Design Automation Conference (DAC), 1994, pp. 81-86
- [15] K. T. Cheng, V. D. Agrawal: "A Partial Scan Method for Sequential Circuits with Feedback", IEEE Trans. on Comp., 1990, pp. 544-548
- [16] R. Gupta, R. Gupta, M. A. Breuer: "The BALLAST Methodology for Structured Partial Scan Design", IEEE Trans. on Comp., 1990, Vol. 39, No. 4, pp. 8-15
- [17] A. Kunzmann and H.-J. Wunderlich: "An Analytical Approach to the Partial Scan Problem", Journal of Electronic Testing: Theory and Application (JETTA) , vol. 1, 1990, pp. 163-174
- [18] E. J. Marinissen, M. Muijen: "SmartScan: Partial Scan with Full Scan Benefits", 4th IEEE International Test Synthesis Workshop, Santa Barbara, CA, 1997
- [19] H.-J. Wunderlich: "The Design of Random-Testable Sequential Circuits", 19th Int. Symp. On Fault-Tolerant Computing (FTCS), 1989
- [20] Journal of Electronic Testing: Theory and Applications (JETTA): Special Issue on Partial Scan Methods, Vol. 7, Aug./Oct. 1995
- [21] R. K. Brayton, G. D. Hachtel, C. McMullen, A. Sangiovanni-Vincentelli: "Logic Minimization Algorithms for VLSI Synthesis", Boston: Kluwer Academic Publishers, 1984
- [22] H.-J. Wunderlich, G. Kiefer: "Bit-Flipping BIST", Proc. Int. Conf. On Computer-Aided Design (ICCAD), 1996, pp. 337-343
- [23] Karp, R. M.: "Reducibility among combinatorial problems"; in R. E. Miller and J. W. Thatcher (eds.), Complexity of Computer Computations, Plenum Press, NewYork, pp.85-103
- [24] F. Brglez, D. Bryan, K. Komzminski: "Combinational Profiles of Sequential Benchmark Circuits", Proc. Int. Symp. On Circuits and Systems (ISCAS), 1989, pp. 1929-1934