Circuit Partitioning for Efficient Logic BIST Synthesis

Alexander Irion[1] Gundolf Kiefer[1] Harald Vranken[2] Hans-Joachim Wunderlich[1]

[1] Computer Architecture Lab
University of Stuttgart
Breitwiesenstr. 20/22, 70565 Stuttgart
Germany
gundolf.kiefer@informatik.uni-stuttgart.de

[2] Philips Research Laboratories
IC Design – Digital Design & Test
Prof. Holstlaan 4, 5656 AA Eindhoven
The Netherlands
harald.vranken@philips.com

Abstract

A divide-and-conquer approach using circuit partitioning is presented, which can be used to accelerate logic BIST synthesis procedures. Many BIST synthesis algorithms contain steps with a time complexity which increases more than linearly with the circuit size. By extracting sub-circuits which are almost constant in size, BIST synthesis for very large designs may be possible within linear time. The partitioning approach does not require any physical modifications of the circuit under test. Experiments show that significant performance improvements can be obtained at the cost of a longer test application time or a slight increase in silicon area for the BIST hardware.

Keywords: circuit partitioning, deterministic BIST, divide-and-conquer

1. Introduction

The use of external automated test equipment (ATE) for testing integrated circuits (ICs) is getting more and more difficult and costly due to increasing pin counts, test data volumes, and clock frequencies, and due to capabilities for testing both digital, analog, memory, and RF modules as integrated into system chips. Built-in self-test (BIST) enables the use of low-cost ATE since requirements on timing accuracy, vector memory, and pin count are strongly reduced. At the moment, BIST for embedded memories is mature and widely used in industry, while industrial use of BIST for random logic is increasing, and BIST for analog and RF modules is emerging.

Logic BIST for industrial applications is currently supported by a few commercial CAD tools [1][2], based on the STUMPS architecture for pseudo-random testing [3] combined with test point insertion for improving test quality [4][5][6]. However, also deterministic logic BIST has been shown to be a viable solution [7][8][9][10][11][12][13][14]. Experimental results in [14] obtained by applying deterministic logic BIST to industrial circuits, show that complete fault coverage is guaranteed at the cost of a small amount of silicon area for the BIST hardware, while the impact on the design process is less when compared to BIST schemes based on test point insertion. In [14] also tradeoffs are reported between test time, test quality, and BIST silicon area: the silicon area can be reduced considerably by applying more test patterns (i.e. longer test time) and/or by decreasing the target fault coverage (i.e. lower test quality).

Tradeoffs in ATE costs, test time, test quality, and BIST silicon area are important factors that affect the costs and quality of each manufactured IC. However, also non-recurring costs related to logic BIST during the design process are important, in particular the computation time used for logic BIST synthesis. For large IC designs, this computation time may become very large, in the order of days or even weeks, which is unacceptable for meeting tight time-to-market windows as typically present for consumer electronics and mobile communication products.

This paper proposes circuit partitioning for reducing the computation time during logic BIST synthesis. By using circuit partitioning, a divide-and-conquer approach is introduced: the circuit is divided into a number of sub-circuits, and logic BIST synthesis is performed for each sub-circuit. Also, the test application is partitioned, so that for each sub-circuit only a relatively small number of test patterns has to be simulated. Experiments show that the sum of the computation times for logic BIST synthesis for all sub-circuits is typically less than the computation time for logic BIST synthesis for the complete circuit in a single run.

Circuit partitioning as a divide-and-conquer approach has been successfully applied in the past on test-related problems like ATPG and fault simulation, e.g. as reported in [15][16][17][18]. In this paper, circuit partitioning is proposed for the first time for reducing time and space complexity of logic BIST synthesis.

The circuit partitioning is used only during BIST synthesis and does not require any physical modifications of the circuit. The proposed approach is orthogonal to core-based design [20]. For core-based systems, BIST is typically applied at the core level, and hence each individual core will have its own BIST

hardware, although sharing of BIST hardware between cores is possible as well. For such core-based systems, in which each core can still be a large circuit, circuit partitioning is applied on the core level.
In this paper, the complexity of BIST synthesis is discussed in Section 2, and the BIST scheme which has been used in [14] is summarized as a case study. The divide step of the circuit partitioning approach is outlined in Section 3, followed by the conquer step in Section 4. Finally, experimental results are presented in Section 5, which show the tradeoffs between BIST synthesis time, silicon area, and test time. Section 6 concludes the paper.

2. The complexity of BIST synthesis

The results reported in [14] were obtained using the Bit-Flipping scheme which is described in detail in [10][11][12][13]. The target structure (see Figure 1) is based on the classical STUMPS scheme. Patterns are generated on-chip using an LFSR. In order to detect all faults (including hard-to-detect ones), a small combinational module, the bit-flipping logic (BFL), is added which modifies the LFSR output sequences at certain predetermined bit positions.

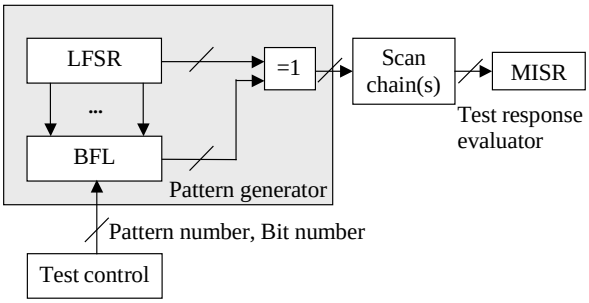


Figure 1: Target structure of a deterministic BIST scheme

During logic BIST synthesis, the BFL is constructed iteratively as sketched in Figure 2. In the beginning of each iteration the current pattern generator is simulated, and by fault simulation the currently undetected faults as well as the essential patterns for the already detected faults are determined. Then deterministic patterns for the undetected faults are computed by an ATPG tool, and the BFL is enhanced such that some of these patterns are generated while the essential old patterns remain unchanged.

Similar to other BIST synthesis schemes, the synthesis procedure includes various time-consuming steps that are called repeatedly, such as logic simulation to simulate the pattern generator hardware and to obtain the MISR's signature, and fault simulation.

The time complexity of logic simulation is linear with the circuit size, i.e. the time for logic simulation of circuit activity in a single clock cycles increases linearly. However larger circuits usually have longer scan chains, and therefore logic simulation may require to simulate more cycles. In addition, larger circuits usually also require more test patterns. Both effects cause that the overall time for logic simulation increases more than linearly with the circuit size.

Fault simulation is among the most expensive tasks in circuit design. Its computing time depends on the size of the circuit, the length of the list of faults to be simulated, and the number of patterns. The average computational complexity for simulating all stuck-at faults for a single pattern turns out to be between quadratic and cubical [21]. Complexity analysis shows that there is no hope for linear time fault simulation [22]. This observation is true for sequential circuits even if they have a pipeline structure. The situation is different for circuits with full-scan design, since the length of critical paths and hence the depth of the purely combinational circuitry is bounded for timing reasons. Modern fault simulation and ATPG tools exploit this fact by an internal partitioning and may handle full-scan designs with nearly linear effort [21][22][23].

All these steps are performed repeatedly, and the number of iterations in the BIST synthesis procedure typically increases with the number of hard-to-detect faults and thus with the circuit size. The total BIST synthesis time may therefore increase more than linearly with the circuit size.

The purpose of this paper is to show that the same kind of partitioning which is successful for fault simulation and ATPG provides significant gains for the BIST synthesis.

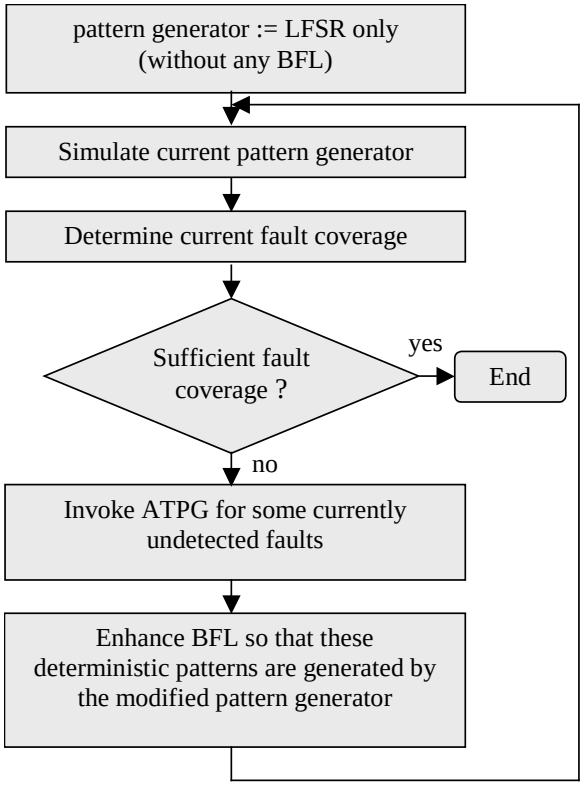


Figure 2: BIST synthesis procedure

Using circuit partitioning, the size of the sub-circuits can be limited to a constant size and the BIST synthesis time for such a sub-circuit is bounded. The BIST synthesis time for a circuit when using partitioning, is approximately equal to the sum of the BIST synthesis times for all sub-circuits. Consequently the BIST synthesis time now increases linearly with the circuit size.

3. Partitioning the circuit

As the divide step of the divide-and-conquer approach a number of sub-circuits is extracted from the circuit under test, such that:

a) Each sub-circuit only contains a subset of the gates and signals of the original circuit, and the size of each sub-circuit is close to a user-specified upper bound.
b) For each testable fault *f* in the original circuit there is a sub-circuit containing the respective gate and signal where the fault is located, and *f* is testable in that sub-circuit.
c) If a test pattern *p* detects some fault *f* in a sub-circuit, then the same fault *f* is detected by the same pattern *p* in the original circuit. For this reason, each sub-circuit includes all the (pseudo-) primary inputs of the complete circuit, so that patterns for sub-circuits are compatible with the original circuit. Obviously the pseudo-primary inputs that do not drive any logic in the sub-circuit are irrelevant during BIST synthesis for the sub-circuit.

Condition a) addresses a desired cost function. Conditions b) and c) ensure that the union of complete test sets for all sub-circuits also achieves complete fault coverage when applied to the original circuit.

The above conditions are satisfied if the sub-circuits are selected based on the output cones of the original circuit as sketched in Figure 3. An output cone of a circuit contains all gates and signals of the transitive fan-in of the output.

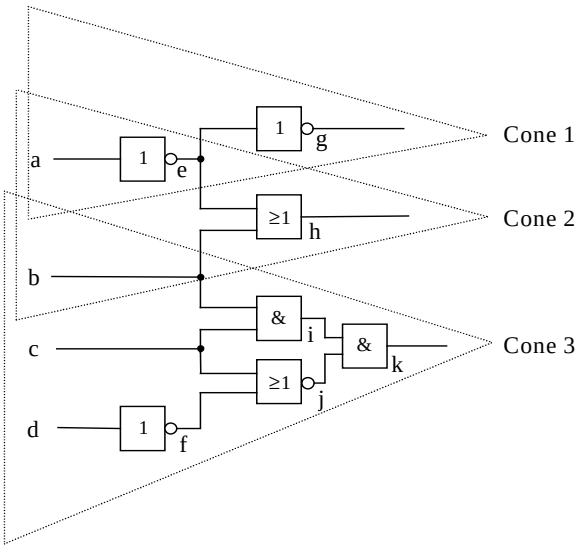


Figure 3: Output cones

Sub-circuits are selected as unions of cones where each cone is contained in exactly one sub-circuit. The sub-circuits are determined by the algorithm as shown in Figure 4.

The cone which has the highest overlap with any other cone is selected as the initial cone. Next, new cones X_i are added to the sub-circuit such that the overlap value

$$overlap(SC,X_i) := \frac{|SC \cap X_i|}{\mathrm{MIN}(|SC|,|X_i|)}$$

is maximum, where $|A|$ is the number of gates in circuit A, and $A \cap B$ is the circuitry that is contained both in A and B. This definition results in a range for the overlap value between 0 (no overlap) and 1 (complete overlap in which one circuit is completely contained in the other circuit). The sub-circuit is constructed iteratively, and in each iteration a new cone is added to the sub-circuit. The loop terminates if the size of the sub-circuit exceeds the user-specified upper bound *MaxSubCircuitSize* and no cone is left that overlaps more than *MinOverlap*.

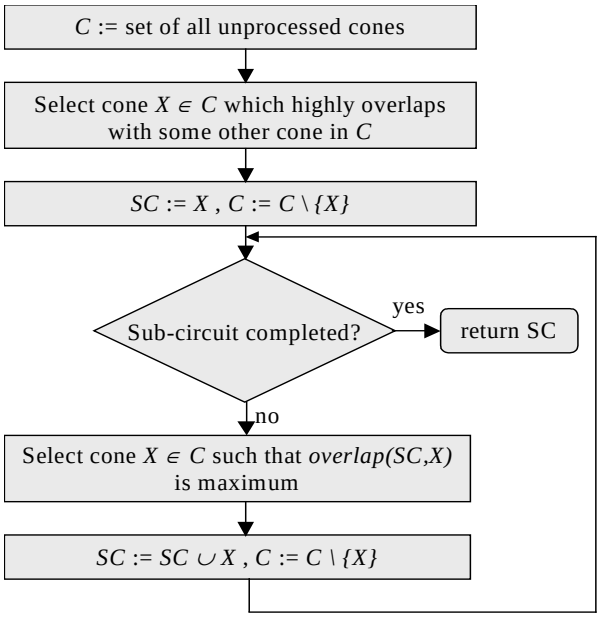


Figure 4: Extracting a sub-circuit for BIST synthesis

Figure 5 shows the BIST synthesis procedure using circuit partitioning. New sub-circuits are extracted as long as the set of cones *C* is not empty. After the BIST synthesis is done for one sub-circuit, an optional full-circuit fault simulation is performed in order to remove cones from the list that are already completely tested.

For example, Figure 6 shows a possible partitioning of the circuit of Figure 3 with *MaxSubCircuitSize* = 5 gates. The circuit is partitioned into two sub-circuits: sub-circuit A contains cones 1 and 2, and sub-circuit B contains cone 3. This example illustrates that the sub-circuits are not necessarily disjoint. However, the algorithm for extracting the sub-circuits as shown in Figure 4 in general provides that the overlap of sub-circuits is small.

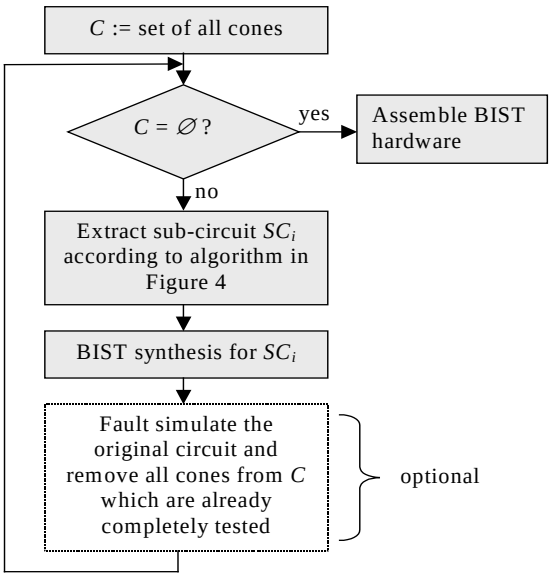
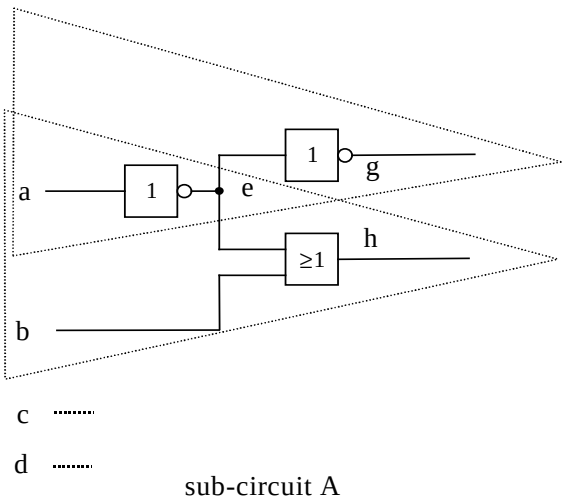


Figure 5: BIST synthesis using circuit partitioning



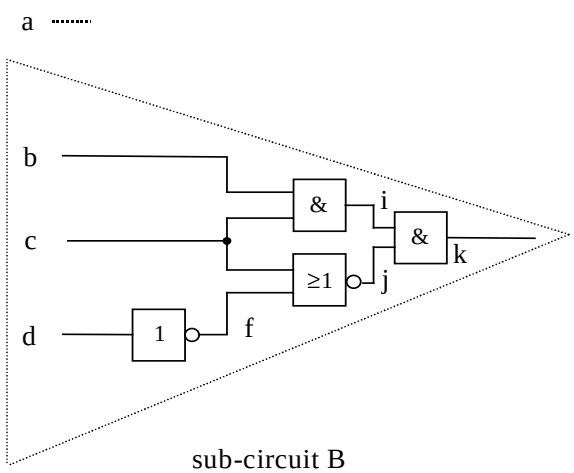


Figure 6: Partitioning of the circuit in Figure 3 into two sub-circuits

4. Assembling BIST hardware for the original circuit

According to the logic BIST synthesis as shown in Figure 5, the partitioning of a circuit into *n* sub-circuits will result in *n* pattern generators, i.e. one pattern generator for each sub-circuit. The BIST hardware for the original circuit is constructed as sketched in Figure 7, containing *n* pattern generators and a global test controller. Let the test application time per pattern generator be 2^k, which results in a test application time of $n \cdot 2^k$ patterns for the complete circuit. The test controller contains a pattern counter with $k + \lceil log_2(n) \rceil$ flip-flops. The lower *k* lines of the pattern counter are connected to the respective pattern generators, e.g. as described in Section 2, while the remaining most significant bits are used to control a multiplexer which selects the output of one of the pattern generators. This architecture implies that the patterns, as generated by the individual pattern generators, are applied to the CUT sequentially.

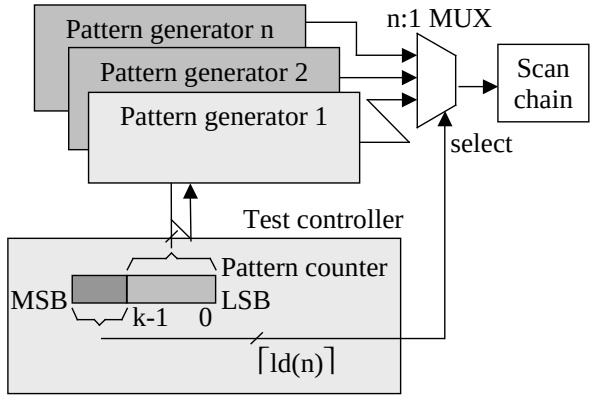


Figure 7: Synthesizing a pattern generator for the original circuit

The structure shown in Figure 7 is general and independent of the underlying BIST scheme. The pattern generators may share logic and may be minimized jointly in the synthesis process. For example, when using the bit-flipping scheme, a single LFSR can be used for all pattern generators, and possibly the *n* BFLs may share logic.

5. Experimental results

We performed a series of experiments with the ISCAS'89 benchmark circuits [19] and industrial circuits from Philips [14]. The ISCAS circuits range in size between 6K to 24K gates, while the circuits from Philips range between 10K and 92K gates. In our initial experiments we determined suitable parameters for *MaxSubCircuitSize* and *MinOverlap*, resulting in *MaxSubCircuitSize* = 2500 nodes and *MinOverlap* = 90% as optimal values. In subsequent experiments we focused on determining tradeoffs between the CPU time of the BIST synthesis procedure, the silicon area for the BIST hardware, and the test application time.

Figure 8 shows the relative sizes of the sub-circuits in one stacked column for each circuit. The size of a sub-circuit is related to the size of the original circuit. Due to the overlapping of the sub-circuits, the sum of all sub-circuits can be more than 100%. On the other hand, the sum can also be lower than 100%, if test patterns for some sub-circuits randomly cover faults of other cones (this is detected by the optional fault simulation step in Figure 5 which we used in our experiments), so that those cones were never extracted and passed to the BIST synthesis tool.

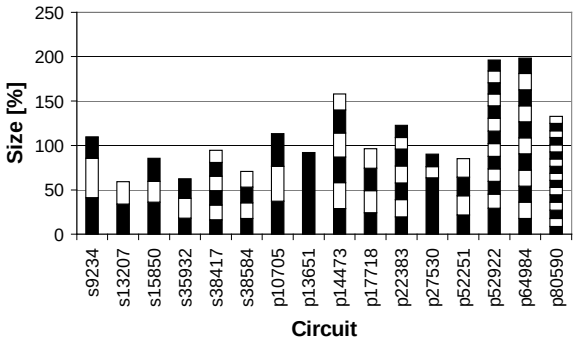


Figure 8: Size of the sub-circuit

Table 1 shows a comparison between the original synthesis procedure without partitioning and the new approach. The test length without circuit partitioning is 16,384 patterns, while the test length with partitioning is 16,384 times the number of generated sub-circuits. The first two columns of Table 1 show the circuit name and the silicon area of the BIST pattern generator synthesized without circuit partitioning using a 1 µm CMOS technology. The right part of the table shows the number of sub-circuits, the absolute size of the combined pattern generator, its relative size with respect to the approach without partitioning, and the relative total computation time of the partitioning procedure with respect to the original non-partitioning approach. The computation time covers the whole BIST synthesis procedure including the partitioning, the BIST synthesis for all sub-circuits, and the final assembly of the BIST hardware.

The rightmost column shows that the BIST synthesis is significantly faster by using the circuit partitioning approach. In 11 out of 16 circuits only less than half the time was required, and in no case a slowdown was observed. The fifth column shows that at the same time the BIST silicon area is roughly the same if the circuit partitioning approach is used: for some circuits we observe a small decrease, and for others a small increase. The circuit p13651 is an exception because a large part of the circuit consists of cones which are mutually highly overlapping. As a consequence, a single sub-circuit was extracted that had almost the size of the original circuit and this sub-circuit therefore was much larger than *MaxSubCircuitSize*. Experiments with different parameter settings, forcing a partitioning into more sub-circuits, resulted in worse computation times for this circuit.

In general, Table 1 shows that without a significant change in silicon area for the BIST hardware, the synthesis procedure can be accelerated considerably. However the price for this is a longer test application time, which is still in an acceptable range if the number of patterns per sub-circuit is e.g. 16,384.

Circuit	Without partitioning	Using circuit partitioning			
	Silicon area[mm^2]	Sub-circuits	Silicon area[mm^2]	Relative silicon area	Relative CPU time
s9234	0.394	3	0.438	111.2%	48.9%
s13207	0.190	2	0.189	99.5%	61.0%
s15850	0.432	3	0.378	87.5%	59.2%
s35932	0.052	3	0.060	115.4%	44.5%
s38417	1.286	6	1.390	108.1%	47.9%
s38584	0.277	4	0.281	101.4%	35.8%
p10705	0.408	3	0.337	82.6%	30.3%
p13651	0.085	1	0.089	104.7%	93.0%
p14473	1.760	6	2.273	129.1%	47.5%
p17718	0.447	4	0.419	93.7%	32.4%
p22383	2.936	7	2.432	82.8%	34.4%
p27530	1.176	3	1.143	97.2%	56.2%
p52251	0.217	4	0.200	92.2%	38.4%
p52922	1.799	13	1.297	72.1%	19.6%
p64984	3.153	11	2.434	77.2%	33.3%
p80590	1.149	16	1.189	103.5%	68.1%

Table 1: Partitioning vs. non-partitioning BIST synthesis with a test length of 16,384 patterns per sub-circuit

In order to get comparable results showing the tradeoff between silicon area and test synthesis time while keeping the test application time constant, we ran the non-partitioning synthesis procedure using a test length which is equal to the test length using partitioning. Increasing the test length for the bit-flipping algorithm of Figure 2 typically increases the BIST synthesis time. However the resulting pattern generator is smaller due to better random fault coverage as well as due to a larger degree of freedom for embedding deterministic patterns and minimizing the BFL. The results are shown in Table 2.

Circuit	Without partitioning	Using circuit partitioning			
	Silicon area[mm^2]	Sub-circuits	Silicon area[mm^2]	Relative silicon area	Relative CPU time
s9234	0.389	3	0.438	112.6%	24.9%
s13207	0.127	2	0.189	148.8%	91.2%
s15850	0.289	3	0.378	130.8%	42.8%
s35932	0.047	3	0.060	127.7%	32.4%
s38417	0.642	6	1.390	216.5%	21.2%
s38584	0.163	4	0.281	172.4%	26.1%
p10705	0.273	3	0.337	123.4%	29.8%
p14473	1.730	6	2.273	131.4%	18.1%
p17718	0.254	4	0.419	165.0%	43.2%
p27530	0.873	3	1.143	130.9%	104.1%
p52251	0.148	4	0.200	135.1%	34.6%

Table 2: Partitioning vs. non-partitioning BIST synthesis with equal test lengths

Compared to the previous experiments, the performance gain during BIST synthesis is even larger: For 7 out of the 11 cases less than 35% of the computation time was needed using circuit partitioning. However, column 5 shows that the size of the pattern generator may increase by 12.6% up to 116.5% in one case.

6. Conclusion

A partitioning approach has been presented which can be used to accelerate BIST synthesis procedures. By extracting sub-circuits of similar sizes, BIST synthesis for very large designs may be possible within linear time.

Without requiring any physical modifications of the circuit under test, experiments have shown that significant performance improvements can be obtained at the cost of a longer test application time or a slight increase in silicon area for the BIST hardware.

Acknowledgements

We thank the various Philips design groups for providing the circuits that we used in our experiments. We also thank Erik Jan Marinissen for his useful suggestions and reviews.

References

[1] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, J. Rajski, *Logic BIST for Large Industrial Designs: Real Issues and Case Studies*, Proceedings International Test Conference, IEEE, 1999, pp. 358-367.

[2] B. Nadeau-Dostie, *Design For At-Speed Test, Diagnosis and Measurement*, Kluwer, 2000.

[3] P.H. Bardell, W.H. McAnney, *Parallel Pseudo-random Sequences for Built-In Test*, Proceedings International Test Conference, IEEE, 1984, pp. 302-308.

[4] J.P. Hayes, A.D. Friedman, *Test Point Placement to Simplify Fault Detection*, IEEE Transactions on Computers, Vol. C-33, July 1974, pp. 727-735.

[5] B.H. Seiss, P.M. Trousborst, M.H. Schulz, *Test Point Insertion for Scan-Based BIST*, Proceedings European Test Conference, IEEE, 1991, pp. 253-262.

[6] N. Tamarapalli, J. Rajski, *Constructive Multi-Phase Test Point Insertion for Scan-Based BIST*, Proceedings International Test Conference, IEEE, 1996, pp. 649-658.

[7] B. Koenemann, *LFSR-Coded Test Patterns for Scan Design*, Proceedings European Test Conference, Munich, 1991, pp. 237-242.

[8] S. Hellebrand, B. Reeb, S. Tarnick, H.-J. Wunderlich, *Pattern Generation for a Deterministic BIST Scheme*, Proceedings ACM/IEEE International Conference on CAD-95 (ICCAD95), San Jose, CA, November 1995, pp. 88-94.

[9] N. A. Touba, E. J. McCluskey, *Altering a pseudo-random bit sequence for scan-based BIST*, Proceedings IEEE International Test Conference, 1996, pp. 167-175.

[10] H.-J. Wunderlich, G. Kiefer, *Bit-Flipping BIST*, Proceedings International Conference on Computer-Aided Design, IEEE, 1996, pp. 337-343.

[11] G. Kiefer, H.-J. Wunderlich, *Using BIST Control for Pattern Generation*, Proceedings IEEE International Test Conference, Washington, DC, November 1997, pp. 347-355.

[12] G. Kiefer, H.-J. Wunderlich, *Deterministic BIST with Multiple Scan Chains*, Proceedings IEEE International Test Conference, Washington, DC, October 1998, pp. 1057-1064.

[13] G. Kiefer, H.-J. Wunderlich, *Deterministic BIST with Partial Scan*, Proceedings IEEE European Test Workshop, Constance, May 25-28, 1999.

[14] G. Kiefer, H. Vranken, E.J. Marinissen, H.-J. Wunderlich, *Application of Deterministic Logic BIST on Industrial Circuits*, Proceedings International Test Conference, IEEE, 2000, pp.105-114.

[15] P.S. Bottorff, R.E. France, N.H. Garges, E.J. Orosz, *Test Generation for Large Logic Networks*, Proceedings Design Automation Conference, IEEE/ACM, 1977, pp. 479-485.

[16] A. Yamada, N. Wakatsuki, T. Fukui, S. Funatsu, *Automatic System Level Test Generation and Fault Location for Large Digital Systems*, Proceedings Design Automation Conference, IEEE/ACM, 1978, pp. 347-352.

[17] S. Patil, P. Banerjee, *Fault Partitioning Issues in an Integrated Parallel Test Generation/Fault Simulation Environment*, Proceedings International Test Conference, IEEE, 1989, pp. 718-726.

[18] R.H. Klenke, R.D. Williams, J.H. Aylor, *Parallelization Methods for Circuit Partitioning Based Parallel Automatic Test Pattern Generation*, Proceedings VLSI Test Symposium, IEEE, 1993, pp. 71-78.

[19] F. Brglez, D. Bryan, K. Komzminski, *Combinational Profiles of Sequential Benchmark Circuits*, Proceedings International Symposium on Circuits and Systems, IEEE, 1989, pp. 1929-1934.

[20] Y. Zorian, E.J. Marinissen, S. Dey, *Testing Embedded-Core Based System Chips*, Proceedings International Test Conference, IEEE, Washington, D.C, October 1998, pp. 130-143.

[21] P. Goel, *Test Generation Costs Analysis and Projections*, Proceedings Design Automation Conference, ACM/IEEE, 1980, pp. 77-84.

[22] D. Harel, B. Krishnamurthy, *Is there hope for Linear Fault Simulation?*, Proceedings International Symposium on Fault-Tolerant Computing, IEEE, 1987, pp. 28-33.

[23] H. Fujiwara, S. Toida, *The Complexity of Fault Detection Problems for Combinational Logic Circuits*, IEEE Transactions on Computers, Vol. C-31, No. 6, June 1982.