Using Tabu Search Method for Optimizing the Cost of Hybrid BIST

Raimund Ubar, Helena Kruus
Tallinn Technical University, Estonia
raiub@pld.ttu.ee, helen.kruus@ttu.ee

Zebo Peng, Gert Jervan
Linköping University, Sweden
{zebpe,gerje}@ida.liu.se

Abstract

This paper deals with a hybrid BIST solution for testing systems-on-chip, which combines pseudo-random test patterns with stored precomputed deterministic test patterns. A method is proposed for finding the optimal balance between pseudorandom and stored test patterns to perform core test with minimum cost of time and memory, and without losing test quality. As a generalization of local optimization, Tabu search method is used for finding the optimal balance. Unlike local search which stops when no improved new solution is found in the current neighborhood, tabu search continues the search from the best solution in the neighborhood even if it is worse than the current solution. To speed up the optimization procedure, a fast method for predicting the location of optimum solution is also used. Experimental results on benchmark circuits have proved the efficiency of the proposed approach for BIST optimization.

1. Introduction

The rapid advances in the areas of deep-submicron technology and design automation tools are enabling engineers to design larger and more complex integrated circuits. These developments are driving engineers toward new System-on-Chip (SOC) design methodologies. SOC is usually designed by embedding predesigned and preverified complex functional blocks, usually referred as cores, into one single die. The cores can be very different by their nature (from analog to memories, including all types of logic) and can be represented in several different ways (RTL code, netlist or layout) [1]. Such a design style allows designers to reuse previous designs and will lead therefore to a shorter time to market and a reduced cost. Testing of SoC, on the other hand, shares all the problems related to testing modern deep submicron chips, and introduces also some additional challenges due to the protection of intellectual property as well as the increased complexity and higher density [2].

To test the individual cores of the system the test pattern source and sink have to be available together with an appropriate test access mechanism (TAM) [3]. A widespread approach implements both source and sink off-chip and requires therefore the use of external Automatic Test Equipment (ATE). But, as the internal speed of SoC is constantly increasing and the technology used in ATE is always one step behind, the ATE solution will soon become unacceptably expensive and inaccurate [4], leading also to an unacceptable yield loss. Therefore, in order to apply at-speed tests and to keep the test costs under control, on-chip test solutions are needed. Such a solution is usually referred to as built-in self-test (BIST).

The BIST architecture demands a test pattern generator (TPG), a test response analyzer (TRA) and a BIST control unit (BCU). The classical way to implement the TPG and TRA for BIST is to use linear feedback shift registers (LFSR). However, the LFSR-based approach often does not guarantee sufficiently high fault coverage (especially in the case of large and complex designs) and demands very long test application times. Therefore, several proposals have been made to combine pseudorandom test patterns, generated by LFSRs, with deterministic patterns [5], [6], [7], [8], [9].

The main concern of mixed-mode BIST approaches has been to improve the fault coverage by mixing pseudorandom vectors with deterministic ones. The issue of cost minimization has not been directly addressed, even though the time minimization problem is discussed in [7].

The main objective of this paper is to find an optimal balance between pseudorandom and deterministic test patterns to perform core test with minimum cost of both time and memory, and without losing in test quality. In [9] two accurate time-consuming algorithms were proposed for finding the optimal time-moment to stop pseudorandom test generation and to apply stored patterns. To speed up the optimization procedure, a method was proposed for approximate estimation of the expected cost for different possible solutions with very low computational overhead.

In this paper, a Tabu search method is implemented for finding the optimal balance between two BIST approaches. Unlike local search which stops when no improved new solution is found in the current neighborhood, tabu search continues the search from the best solution in the neighborhood even if it is worse than the current solution. To find out a better initial solution for Tabu search, a fast method proposed in [9] for predicting the optimum solution is used.

The paper is organized as follows. In Section 2 we describe the concept of hybrid BIST, Section 3 gives an overview of test cost calculation for hybrid BIST together with optimal cost prediction method, in Section 4 we present Tabu search method for optimization the cost of hybrid BIST, and in Section 5 we present the experimental results which demonstrate the feasibility and efficiency of our approach. Finally, in Section 6 we draw some conclusions together with an introduction to future work.

2. Hybrid BIST

As test patterns, generated by LFSRs, are pseudorandom by their nature, the generated test sequences are usually very long and not sufficient to detect all the faults. To avoid the test quality loss due to random pattern resistant faults and in order to speed up the testing process, we have to apply deterministic test patterns targeting the random resistant and difficult to test faults. This hybrid BIST approach starts with on-line generation of pseudorandom test sequence with a length of L. On the next stage, stored test approach takes place. For the stored approach, precomputed test patterns, stored in the memory, are applied to the core under test to reach 100% fault coverage. For off-line generation of S deterministic test patterns (the number of stored test patterns), arbitrary software test generators may be used, based on deterministic, random or genetic algorithms.

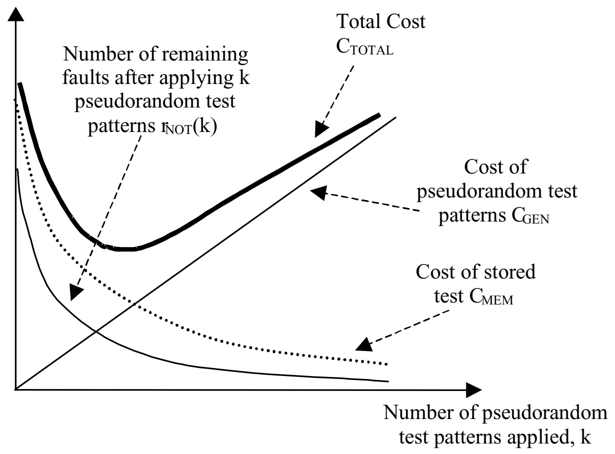

Fig. 1. Cost calculation for hybrid BIST

The length L of the pseudorandom test is an important parameter, which determines the behavior of the whole test process. A shorter pseudorandom test set implies a larger deterministic test set. This however requires additional memory space, but at the same time, shortens the overall testing time. A longer pseudorandom test, on the other hand, will lead to longer test application time with reduced memory requirements. Therefore it is crucial to determine the optimal length of pseudorandom test in order to minimize the total testing cost.

Figure 1 illustrates graphically calculation of the total cost of the hybrid BIST consisting of pseudorandom test patterns and stored test patterns, generated off-line.We can define the total test cost of the hybrid BIST C_{TOTAL} as:

$$C_{TOTAL} = C_{GEN} + C_{MEM} = \alpha L + \beta S$$

where C_{GEN} is the cost related to the time for generating L pseudorandom test patterns (number of clock cycles), C_{MEM} is the memory cost for storing S precomputed test patterns (number of stored test patterns), and α, β are constants to map the test length and memory space to the costs of the two parts of the test solutions to be mixed.

Figure 1 describes how the cost C_{GEN} of pseudorandom test is increasing when striving to higher fault coverage. In general, it can be very expensive to achieve high fault coverage with pseudorandom test patterns only. The curve C_{MEM} describes the cost that we have to pay for storing precomputed tests at the given fault coverage reached by pseudorandom testing. The total cost C_{TOTAL} is the sum of the two mentioned costs αL and βS. The weights α and β reflect the correlation between the cost and the pseudorandom test time (number of clock cycles used) or between the cost and the memory size needed for storing the precomputed test sequence. For simplicity we take here $\alpha = 1$, and $\beta = B$ where B is the number of bytes of the input test vector to be applied on the core under test (CUT). Hence, in the following we will use as the cost units, number of clocks used for pseudorandom test generation and number of bytes in the memory needed for storing the precomputed test patterns.

3. Cost Calculation for Hybrid BIST

The main purpose of this work is to find a fast method for finding the length L of the pseudorandom circuit when the total cost C_{TOTAL} has the minimal value C_{min}. Creating the curve $C_{GEN} = \alpha L$ is not difficult. For that purpose, a simulation of the behavior of the LSFR, a pseudorandom test pattern generator (PRG), is needed, and a fault simulation should be carried out for the complete test sequence generated by the LFSR. As a result of such a simulation, we find for each clock cycle the list of faults, which were covered at this clock cycle.

In Table 1 a fragment of the results of BIST simulation for the ISCAS'85 circuit c880 [10] is demonstrated. Here

- k is the number of the clock cycle,
- $r_{DET}(k)$ is the number of new faults detected by the test pattern generated at the clock signal k,
- $r_{NOT}(k)$ is the number of faults not yet covered by the sequence generated by k clock signals,
- $FC(k)$ is the fault coverage reached by the sequence of patterns generated by k clock signals

k	$r_{DET}(k)$	$r_{NOT}(k)$	$FC(k)$
0	155	839	15.593561%
1	76	763	23.239437%
2	65	698	29.778671%
3	90	608	38.832996%
4	44	564	43.259556%
5	39	525	47.183098%
10	104	421	57.645874%
15	66	355	64.285713%
20	44	311	68.712273%
28	42	269	72.937622%
50	51	218	78.068413%
70	57	161	83.802818%
100	16	145	85.412476%
148	13	132	86.720322%
200	18	114	88.531189%
322	13	101	89.839035%
411	31	70	92.957748%
707	24	46	95.372231%
954	18	28	97.183098%
1535	4	24	97.585510%
1560	8	16	98.390343%
2153	11	5	99.496979%
3449	2	3	99.698189%
4519	2	1	99.899399%
4520	1	0	100.000000%

Table 1. Pseudorandom test results

In the list of BIST simulation results not all clock cycles should be represented. Only these clock numbers are interesting at which at least one new fault will be covered, and the total fault coverage for the pseudorandom test sequence up to this clock number increases. Let us call such clock numbers and the corresponding pseudorandom test patterns *resultative clocks* and *resultative patterns*.

More difficult is to find the values for $C_{MEM} = \beta S$. Let $t(k)$ be the number of test patterns needed to cover $R_{NOT}(k)$ not yet detected faults (these patterns should be precomputed and used as stored test patterns in the hybrid BIST). Calculation of $t(k)$ is the most expensive procedure. In [9], two algoritms were suggested for calculating $t(k)$ based either using an ATPG for generating test patterns for each k, or based on the manipulations with fault tables. In case of very large circuits both of these algorithms will lead to expensive and time-consuming experiments.

4. Cost optimization with Tabu search method

Tabu search was introduced by Fred Glover [11], [12], [13] as a general iterative heuristic for solving combinatorial optimization problems. Initial ideas of this technique were also proposed by Hansen [14] in his *steepest ascent descent* heuristic.

```
Begin
Start with initial solution $S \subset \Omega$;
BestSolution:=S;
Add move to S to tabu list;
While number of empty iterations < E Or
 there is no return to previously visited solution Do
    Generate the sample of neighbor solutions
    $V^* \subset N(S)$;
    Find best $Cost(S^* \subset V^*)$;
M:  If move to solution S* is not T Then
        $S^{trial}$:=S*;
        Update tabu list;
        Increment the iteration number;
    Else
        Find the next best $Cost(S^* \subset V^*)$;
        Go to M;
    EndIf
    If $Cost(S^{trial}) < Cost(BestSolution)$ Then
        BestSolution := $S^{trial}$;
    Else
        Increment number of empty iterations
    EndIf
EndWhile
End.
```

Fig. 2. Tabu search algorithm

Tabu search is a form of local neighborhood search. Each solution $S \in \Omega$ where Ω is the search space (the set of all feasible solutions) has an assosiated set of neighbours $N(S) \subseteq \Omega$. A solution S'

∈ N(S) can be reached from S by an operation called a *move* to S'. At each step, the local neighborhood of the current solution is explored and the best solution is selected as a new current solution. Unlike local search which stops when no improved new solution is found in the current neighborhood, tabu search continiues the search from the best solution in the neighborhood even if it is worse than the current solution. To prevent cycling, information pertaining to the most recently visited solutions are inserted in a list called *tabu list*. Moves to the tabu solutions are not allowed. The tabu status of a solution is overridden when a certain criteria (aspiration criteria) is satisfied. One example of an aspiration criterion is when the cost of the selected solution is better than the best seen so far, which is an indication that the search is actually not cycling back, but rather moving to a new solution not encountered before [11].

The procedure of the tabu search starts from an initial feasible solution S (current solution) in the search space Ω. A neighborhood N(S) is defined for each S. A sample of neighbor solutions V* ⊂ N(S) is generated. An extreme case is to generate the entire neighborhood , that is to take V* = N(S). Since this is generally impractical (computationally expensive), a small sample of neighbors (V* ⊂ N(S)) is generated called *trial* solutions (│V* │= n << │N(S)│). From these trial solutions the best solution say S*∈ V*, is chosen for consideration as the next solution. The move to S* considered even if S* is worse than S, that is, *Cost(S*)>Cost(S)*. A move from S to S* is made provided certain conditions are satisfied.The best candidate solution S*∈ V* *may* or *may not* improve the current solution but is still considered. It is this feature that enables *escaping* from local optima.

One of the parameters of the algorithm is the size of the tabu list. A tabu list T is maintained to prevent returning to previously visited solutions. The list contains information that to some extent forbids the search from returning to a previously visited solutions. Generally the tabu list size is small. The size can be determined by experimental runs, watching the occurance of cycling when the size is too small, and the deteriation of solution quality when the size is too large [15].

Let's have the following additional notations: E - number of allowed empty iterations (i.e. iterations that do not result in finding a new best solution), defined for each circuit, and S^{trial} - solution generated from current solution as a result of the move. The Tabu search based algorithm implemented in this paper is represented on Fig. 2.

For finding the initial feasible solution to make Tabu search more productive, a fast estimation method for optimum L proposed in [9] is used. For this estimation, the number of not yet covered faults in $R_{NOT}(i)$ can be used. The value of $|R_{NOT}(i)|$ can be acquired directly from the PRG simulation results and is available for every significant time moment (see Table 1). Based on the value of $|R_{NOT}(i)|$ it is possible to estimate the expected number of test patterns needed for covering the faults in $R_{NOT}(i)$. The starting point for the Tabu search procedure can be found by giving rough estimation of the total cost based on the value of $|R_{NOT}(i)|$. Based on statistical analysis of the costs calculated for ISCAS'85 benchmark circuits, in [9] the following coefficient was chosen: 1 remaining fault = 0,45 test patterns needed to cover it. In this way, a simplified cost prediction function was derived: $C'_{TOTAL}(k) = C_{GEN}(k) + 0{,}45\beta \cdot R_{NOT}(k)$ whereas the value of k^* where $C'_{TOTAL}(k^*) = min$ over k was used for theinitial solution for Tabu search.

5. Experimental results

Experiments were carried out on ISCAS'85 benchmarks in order to demonstrate the advantage of Tabu search compared to the known methods. Investigations were carried out to find the best initial solution, the step defining *N(S)*, the size of V* and the size of tabu list for using the Tabu strategy in a most efficent way.

For finding the best initial solution the cost prediction method proposed in [9] was used. For finding the tabu list size, experiments were carried out with different sizes of the list. Results showed that the best average size for the ISCAS'85 benchmark family was 3. Smaller list size would cause cycling around local minimum, larger size would result in deterioration of solution quality. The size of the sample of neighborhood solutions V* giving the best results for all circuits, was 4. Smaller size would make the process of finding the minimum very long, resulting in very small speedup. Larger size of V* did not improve the results.

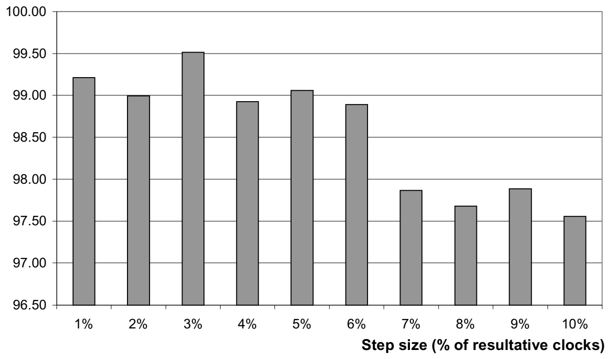

Fig. 3. Dependency of estimation accuracy from neighborhood step size

The efficiency of search depends significally of the step size defining the neighborhood N(S). Based on the experimental results, the charts of dependancy of the overall estimation accuracy and of the overall speedup from step size were compozed. Analysing results depicted in Fig. 3 and Fig. 4 led to the conclusion that the most admissable step size can be counted as 3% of resultative clocks, where the average estimation accuracy is the highest. Though the larger step size would give us the increase of the speedup, it was found inadmissable because of the rapid decrease in the cost estimation accuracy.

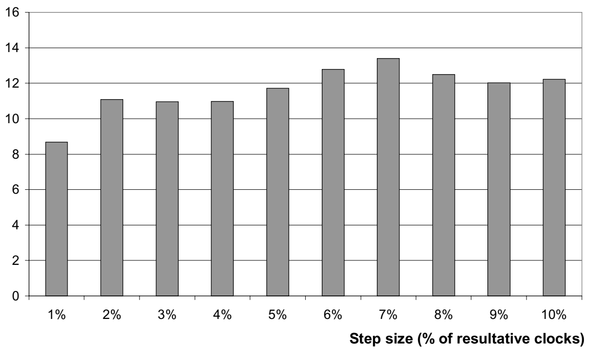

Fig. 4. Dependency of average speedup from neighborhood size

	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552
Simulated clocks	780	2036	5589	1522	5803	6581	8734	2318	210	18704
Resultative clocks	81	114	114	109	183	118	265	252	53	279
Actual total cost	165	398	366	374	487	2397	771	1072	63	3203
Estimated total cost	165	398	367	376	487	2420	771	1103	63	3213
Estimation accuracy, %	100.00	100.00	99.73	99.47	100.00	99.05	100.00	97.19	100.00	99.69
Number of calculations	11	19	15	18	28	9	16	12	15	8
Speedup	7.36	6.00	7.60	6.06	6.54	13.11	16.56	21.00	3.53	34.87
Iterations made	7	14	10	13	17	6	10	9	12	5

Table 2. Experimental results

Investigations were carried out to find out the criteria to stop iterations. Analyzing the experiments with a fixed number of iterations to find out which number of empty iterations E (iterations not giving new best cost) will most probably end up in getting no better solution, we found the value E = 7. The other criteria to stop iterations is the return of the search to previously visited solution.

The results of optimization of the hybrid BIST for the ISCAS'85 benchmark circuits obtained using parameters described above are depicted in Table 2. The number of cost calculations is given in row 6, the total number of iterations in row 8.

6. Conclusions and future work

This paper describes a hybrid BIST solution for testing systems-on-chip. It combines pseudorandom test patterns with deterministic test patterns. For selecting the optimal switching moment from pseudorandom test generation mode to the stored deterministic patterns mode Tabu search method was used to search for a global minimum in a search space consisting of a lot of local minimums.

The use of Tabu search method allowed to reduce significantly the number of exact calculations of the cost function compared ot the known methods. The speedup varies from 3 to 25 times whereas the accuracy of the solution (the reached minimum cost compared to the exact minimum) was not less than 97,2 % for the whole family of ISCAS'85 benchmark circuits.

In such a way, the experimental results demonstrate that the proposed approach is feasible and efficient for finding optimized solutions for hybrid BIST architectures that are used as the most important test conception in systems-on-chip.

As a future work we would like to investigate possibilities to use the proposed approach for parallel testing issues (testing multiple cores at the same time) and to use the same ideas in case of sequential cores.

Acknowledgements

The authors appreciate the work of Jaan Raik and Elmet Orasson from Tallinn Technical University for developing software tools. The work was supported by EC Copernicus project VILAB, the Swedish National Board for Industrial and Technical Development (NUTEK), and the Estonian Science Foundation (Grants No 3658, 4003).

References

[1] "Virtual Socket Interface Architectural Document," VSI Alliance, Nov. 1996.

[2] E. J. Marinissen, Y. Zorian, "Challenges in Testing Core-Based System ICs," *IEEE Communications Magazine,* pp. 104-109, June 1999.

[3] Y. Zorian, E. J. Marinissen, S. Dey, "Testing Embedded Core-Based System Chips," *IEEE International Test Conference (ITC),* pp. 130-143, Washington, DC, October 1998. IEEE Computer Society Press.

[4] "The National Technology Roadmap for Semiconductors," Semiconductor Industry Assoc., San Jose, Calif., 1997, 1998.

[5] S. Hellebrand, S. Tarnick, J. Rajski, B. Courtois, "Generation Of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," *IEEE Int. Test Conference (ITC'92),* pp. 120-129, Baltimore, 1992.

[6] S. Hellebrand, H.-J. Wunderlich, A. Hertwig, "Mixed-Mode BIST Using Embedded Processors," *Journal of Electronic Testing: Theory and Applications*, pp. 127-138, No. 12, 1998.

[7] M. Sugihara, H. Date, H. Yasuura, "Analysis and Minimization of Test Time in a Combined BIST and External Test Approach," *Design, Automation & Test In Europe Conference (DATE 2000)*, pp. 134-140, Paris, France, March 2000.

[8] N. Zacharia, J. Rajski, J. Tyzer, "Decompression of Test Data Using Variable-Length Seed LFSRs," *13th VLSI Test Symposium,* pp. 426-433, 1995.

[9] G.Jervan, Z.Peng, R.Ubar. Test Cost Minimization for Hybrid BIST. *IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems*. Tokio, October 25-28, 2000, pp.283-291.

[10] F. Brglez, P. Pownall, R. Hum. "Accelerated ATPG and fault grading via testability analysis," *IEEE Int. Symp. on Circuits and Systems,* pp. 663-698, June 1985.

[11] F.Glover and M.Laguna. *Tabu Search*. Kluver, MA, 1997.

[12] F.Glover. Tabu search - Part I. ORSA *Journal on Computing*. 1(3): 190-206, 1989.

[13] F.Glover, E.Taillard, and D. de Werra. A user's guide to tabu search. *Annals of Operations Research*, 41:3-28, 1993.

[14] P.Hansen. The steepest ascent mildest descent heuristic for combinational programming. *Congress on Numerical Methods in Combinatorial Optimization*, 1986.

[15] Sadiq M. Sait, Habib Youssef. "Iterative Computer Algorithms with Application in Engineering. *Solving Combinatorial Optimization Problems.*" 1999.