Hybrid BIST Optimization for Core-based Systems with Test Pattern Broadcasting

Raimund Ubar, Maksim Jenihhin
Department of Computer Engineering
Tallinn Technical University, Estonia
{raiub, maksim}@pld.ttu.ee

Gert Jervan, Zebo Peng
Embedded Systems Laboratory (ESLAB)
Linköping University, Sweden
{gerje, zebpe}@ida.liu.se

Abstract[1]

This paper introduces a technique for hybrid BIST time optimization for testing core-based systems that use test pattern broadcasting for both pseudorandom and deterministic patterns. First we formulate the test time minimization problem for such an architecture. Thereafter we present algorithms for finding an efficient combination of pseudorandom and deterministic test sets under given memory constraints, so that the system testing time can be shortened. We also analyze the significance of the pseudorandom sequence quality for the final results. The results are illustrated and the efficiency of the approach is demonstrated by experimental results.

1. Introduction

Built-in self-test (BIST) has became increasingly viable solution for testing complex systems-on-chip (SoC). Although it is a promising technology it also has its problems. Some of the problems are related to the fact that BIST uses typically long sequences of pseudorandom test patterns that may lead to the very long test times and cannot guarantee sufficiently high fault coverage. Therefore we have proposed hybrid BIST [1], [2] as a possible improvement of a classical logic BIST for testing SoCs. The key issue for the hybrid BIST is to find the best balance between pseudorandom and deterministic test patterns, such that the system design constraints are satisfied and test cost is minimized.

Testing core-based systems is a complex problem in general. The existing work has concentrated so far on test scheduling, TAM design and testability analysis. This assumes a fixed set of tests and test resources together with an appropriate test access architecture. Some approaches can also take into account test conflicts and different constraints, e.g. power [3]-[8]. However there hasn't been any work to find the optimal test sets for testing every individual core in such a manner that the total system test time is minimized and the different design constraints are satisfied.

In our earlier work it has been assumed that every core has its own dedicated BIST logic that is capable to produce a set of independent pseudorandom test patterns [9]. We have also extended the same approach for multi-core systems where both, combinatorial cores and sequential cores with full scan may be used [2], [10]. This however may lead to high area overhead and may require redesign of the cores as not all cores may be equipped with self-test structures. Therefore we have recently proposed a novel self-test architecture that is based on test pattern broadcasting [11]. In this approach only a single pseudorandom test pattern generator is used and all test patterns are broadcasted simultaneously for all cores in the system. These patterns will be complemented with dedicated deterministic patterns for every individual core, if needed. Those deterministic test vectors are generated during the development process and are stored in the system.

The whole test process has to be carried out in such a manner that the total testing time is kept minimal without violating the design constraints, in particular, the amount of on-chip resources. In this paper we will propose a method to evaluate tradeoffs between the length of the pseudorandom test sequences and the number of stored deterministic patterns, under given memory constraints. The problem of finding the exact solution is NP-complete. To overcome the high complexity of the problem we will propose in the following a simple and fast algorithm that gives us a near-optimal solution with low computational cost. Although the solution is not optimal it can be used successfully for design space exploration and gives a significant improvement compared to the ad-hoc designer solution.

In the following section we will formulate the test time minimization problem for hybrid BIST. It is followed by

[1] This work has been supported by the EC project EVIKINGS (IST-2001-37592), Estonian Science Foundation Grants 4300 and 5649, and by the Swedish Foundation for Strategic Research (SSF) under the Strategic Integrated Electronic Systems Research (STRINGENT) program.

the proposed test time minimization algorithm that is followed by experimental results and conclusions.

2. Formulation of the test time minimization problem

Let us assume a system S, consisting of cores C_1, C_2, …, C_n. For this system a pseudorandom test sequence TP with length LP is generated and applied in parallel to all cores. This sequence should preferably achieve 100% fault coverage for all cores. In this sequence we can specify subsequences TP_k with length LP_k, $k = 1, 2, …, n$, for each core, so that all the subsequences start in the beginning of TP, and by the last pattern of a subsequence TP_k the 100% fault coverage for the core C_k is reached.

In a case when LP_k is too long, we restrict the length of the pseudorandom sequence to the maximum acceptable length LP_{max}, thus reducing the length of the whole pseudorandom sequence to LP_{max}. For all cores where 100% fault coverage has not been achieved with this test set TP we generate complementary joint set of deterministic test patterns TD, so that by applying to the system both test sequences TP and TD with total length L, the 100% fault coverage for all cores is achieved.

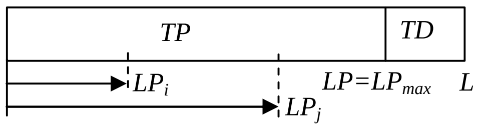

Figure 1. Initial test sequence for multi-core system

As an example, in Figure 1 a hybrid test sequence $TH = \{TP, TD\}$ is shown consisting of a pseudorandom test set TP with length LP and a deterministic test set TD with length LD $(L=LP+LD)$. Here LP_i denotes a moment where 100% fault coverage is reached for the core C_i, and LP_j denotes a moment where 100% fault coverage is reached for the core C_j. In this example we assume that not for all cores 100% fault coverage is achieved by the pure pseudorandom test sequence TP and an additional deterministic test set TD has to be applied to achieve 100% fault coverage. Those deterministic test patterns are precomputed and stored in the system.

The main problem of the hybrid BIST is to find the optimal balance between the pseudorandom test part TP and the deterministic test part TD, so that the total testing time is minimal, and that the memory constraints $COST_{M.LIMIT}$ for storing deterministic test patterns are satisfied, $COST_M \le COST_{M,LIMIT}$. The memory cost can be calculated as follows:

$$COST_M = \sum_{k=1}^{n} (LD_k * INP_k),$$

where INP_k is the number of inputs of the core C_k and LD_k is the length of the deterministic test set of the core C_k. If the same deterministic pattern is needed simultaneously for a subset $S' \subseteq S$ of cores, we say that it is dedicated for the core $C_k \in S'$ with the highest number of inputs.

The task to be solved in this paper is to minimize the total test length

$$LH = LP + \sum_{k=1}^{n} LD_k$$

for a given memory constraint $COST_M \le COST_{M,LIMIT.}$

As all cores are tested in parallel, the problem is to find a time moment when to switch from the parallel pseudorandom test to the parallel deterministic test. The problem of minimizing the hybrid test length at the given memory constraints for parallel multi-core testing is extremely complex. The main reasons of this complexity are the following:

- The deterministic test patterns of one core are used as pseudorandom test patterns for all other cores; unfortunately there will be $n(n-1)$ relationships for n cores to analyse for finding the optimal interaction; on the other hand the deterministic test sets are not readily available and calculated only during the analysis process;
- For a single core an optimal combination of pseudorandom and deterministic patterns can be found by rather straightforward algorithms [9]; but as the optimal time moment for switching from pseudorandom to deterministic test will be different for different cores the existing methods cannot be used and the parallel testing case is considerably more complex.
- For each core the best initial state of the LFSR can be found experimentally, but to find the best LFSR for testing all cores in parallel is a very complex and time consuming task.

To overcome the high complexity of the problem we will propose a straightforward algorithm for calculating TP and TD, where we neglect the optimal solutions for individual cores in favour of finding a near-optimal solution for the whole system.

3. Test time minimization procedure

We solve the test time minimization problem in three consecutive steps: first, we find as good as possible initial state for the LFSR for all cores; second, we generate a deterministic test sequence if the 100% fault coverage cannot be reached by a pure pseudorandom test sequence for all cores; and third, we update the test sequence by finding the quasi-optimal time moment for switching from parallel pseudorandom testing to parallel deterministic testing at the given memory constraint.

Finding the initial state for the LFSR.

To find the best initial state for the parallel pseudorandom test generator, we carry out m experiments, with randomly chosen initial states, for all n cores. Within each j^{th} experiment we calculate for each core C_k the weighted length $LP_{k,j} * INP_k$ of the test sequence which achieves the 100% fault coverage for the core C_k. Then, for all the experiments we calculate the average weighted length

$$L_j = \frac{1}{n}\sum_{k=1}^{n} LP_{k,j} * INP_k$$

as the quality merit of pseudorandom sequences for parallel testing of all cores. The best pseudorandom sequence is the one that gives as shortest L_j, $j = 1,2,\ldots, m$. Let us call this initial pseudorandom test TP^0.

Generation of the initial deterministic test set.

Suppose there are $k \leq n$ cores where 100% fault coverage cannot be achieved with TP^0 because of the practical constraints to the pseudorandom test length. Let us denote this subset of cores with $S' \subseteq S$. Let us denote with FP_i^0 fault coverage of the core C_i, achieved by TP^0. Let us order the cores in S' as $C_1, C_2, \ldots, C_k$, so that for each $i < j$, $1 \leq i,j \leq k$, we have $FP_i \leq FP_j$. We assume here that every deterministic test pattern, to be propagated to the system, has to be as wide as the maximum width of the TAM. If the core under test has less inputs than the width of the TAM, all unused bits in the TAM are filled with pseudorandom data. The deterministic patterns can be generated by using the following algorithm:

Algorithm 1.

1. Start with core C_i in S', i=1.
2. Generate a deterministic test set TD'_i to complement the TP^0 to increase the fault coverage FP_i^0 of the core C_i to 100%.
3. Fill the unused bits of TD'_i with pseudorandom data by continuing the pseudorandom test TP^0. Denote this updated test by TD_i.
4. Broadcast the test TD_i for other cores in S', fault simulate it for the cores in S', and update the fault coverage FP_j^0 for other cores in S'.
5. Take the next core C_i in S' for $i = i + 1$.
6. If $i > k$, END.
7. If $FP_i^0 = 100\%$, repeat Step 5, else go to Step 2.

By using Algorithm 1 an initial hybrid BIST sequence $TH^0 = \{TP^0, TD^0\}$ can be generated. This sequence guarantees 100% fault coverage for all cores in the system.

Definition 1: A pattern in a joint pseudorandom test sequence is called *efficient* if it detects at least one new fault for at least one core that is not detected by previous test patterns in the sequence nor by any pattern in the deterministic test sequence.

Optimization of the test sequence.

After the previous 2 steps we have obtained a hybrid BIST sequence $TH^0 = \{TP^0, TD^0\}$ with length LH^0, consisting of the pseudorandom part TP^0 with length LP^0, and of the deterministic part TD^0 with length LD^0.

In special case TD^0 may be an empty set.

Let us denote with $COST_M(TD^0)$ the memory cost of the deterministic test set TD^0. We assume that the memory constraints are at this moment satisfied: $COST_M(TD^0) < COST_{M,LIMIT}$. In a opposite case, if $COST_M(TD^0) > COST_{M,LIMIT}$, the length of the pseudorandom sequence has to be extended and the second step of the procedure has to be repeated.

If $COST_M(TD^0) = COST_{M,LIMIT}$ the third step is unnecessary, and the procedure is finished.

Under optimization of TH^0 we mean the minimization of the test length LH^0 at the given memory constraints $COST_{M,LIMIT}$.

It is possible to minimize LH^0 by shortening the pseudorandom sequence, i.e. by moving step-by-step efficient patterns from the beginning of TP^0 to TD^0 and by removing all other patterns between the efficient ones from TP^0, until the memory constraints will become violated, $COST_M(TD^0) > COST_{M,LIMIT}$.

We cannot remove patterns with the same goal from the other end of TP^0 because the pseudorandom sequence will be extended and merged with the deterministic part TD^0 to update the free bits of deterministic test patterns generated by Algorithm 1. In other words, by removing pseudorandom patterns from the end of the TP^0 would brake the continuity of the pseudorandom test generation process on the border between TP^0 and TD^0.

To find the efficient test patterns in the beginning of the TP^0 we have to fault simulate the whole test sequence TH^0 for all the cores in the opposite way from the end to the beginning. As a result of the fault simulation we get for each pattern the increments of fault coverage in relation to each core $D = \{D_1, D_2, \ldots, D_n,\}$. According to Definition 1, we call the pattern *efficient* if

$$\exists k, k = 1,2,\ldots,n : \Delta_k \neq 0$$

The optimization procedure will be carried out by using the following algorithm.

Algorithm 2.

1. Start with the first pattern P_i from the beginning of TP^0, $i = 1$.
2. If P_i is efficient, move it from TP^0 to TD^0.
3. Recalculate the memory cost $COST_M(TD^0) = COST_M(TD^0) + COST_M(P_i)$.

4. If $COST_M(TD^0) < COST_{M,LIMIT}$ go to Step 5,
   else if $COST_M(TD^0) > COST_{M,LIMIT}$ go to Step 7,
   else go to Step 8.
5. Take the next pattern P_i in TP^0, $i = i + 1$.
6. If P_i is not efficient, remove it from TP^0, and go to Step 5;
   else go to Step 2.
7. Remove P_i from TD^0 back to TP^0. Go to 10.
8. Take the next pattern P_i in TP^0, $i = i + 1$.
9. If P_i is not efficient, remove it from TP^0, and go to Step 8.
10. END: take P_i as the new beginning of the pseudorandom test sequence TP^0 .

As the result of the Algorithm 2 we create a new hybrid BIST sequence $TH = \{TP,TD\}$ with total length LH and with lengths $LP \leq LP^0$ and $LD \geq LD^0$ for the new pseudorandom and deterministic parts correspondingly. Due to removal of all non-efficient patterns $LP - LP^0 >> LD^0 - LD$. Hence, the total length of the new hybrid BIST sequence will be considerably shorter compared to its initial length, $LH < LH^0$.

The memory constraints, according to the Algorithm 2, remain satisfied: $COST_M(TD) < COST_{M,LIMIT}$.

The described procedure doesn't guarantee absolute minimum of the test length, however, the procedure is rather straightforward (similar to the greedy algorithm) and fast and therefore suitable for use in the design process. The method can be used to find a cheap practical solution as well as for a fast reference for comparison with more sophisticated optimization algorithms to be developed in the future.

4. Experimental data

We have performed experiments with three systems composed from different ISCAS benchmarks as cores. The data of these systems are presented in Table 1 (the lists of used cores in each system)

To show the importance of the first step of the procedure, i.e. the significance of the quality of the initial state of the LFSR, a comparison of the best and worst initial states of the LFSR for all 3 experimental systems has been carried out. The lengths of a complete pseudorandom test sequence (100% fault coverage), starting from the best and worst initial state, are depicted in Table 2. In case of system S3 the pseudorandom sequence was unacceptably long. Therefore the pseudorandom test generation was interrupted and an initial set of deterministic test patterns was generated in order to achieve 100% fault coverage.

System name	**S1** 6 cores	**S2** 7 cores	**S3** 5 cores
List of used cores	c5315	c432	c880
	c880	c499	c5315
	c432	c880	c3540
	c499	c1355	c1908
	c499	c1908	c880
	c5315	c5315	
		c6288	

Table 1. Systems used for experiments

System Name	The best initial state for the pseudorandom test		The worst initial state for the pseudorandom test	
	Pseudorandom test length (clocks)	Deterministic test length (clocks)	Pseudorandom test length (clocks)	Deterministic test length (clocks)
S1	2 520	0	23 482	0
S2	7 060	0	23 482	0
S3	14 524	26	25 000	33

Table 2. Quality of different pseudorandom sequences

The experimental results for three different systems are presented in Table 3. The lengths of the pseudorandom test sequence, the number of additional deterministic test patterns and the total length of the hybrid test sequence is calculated for three different memory constraints and for

System Name	Number of cores	Memory Constraint (bits)	The best initial state for the pseudorandom test				The worst initial state for the pseudorandom test			
			Pseudorandom test length (clocks)	Deterministic test length (clocks)	Total test length (clocks)	CPU time (sec)	Pseudorandom test length (clocks)	Deterministic test length (clocks)	Total test length (clocks)	CPU time (sec)
S1	6	20 000	85	181	266	187, 64	2 990	138	3128	228.67
		10 000	232	105	337		4 446	73	4519	
		5 000	520	55	575		5 679	40	5719	
S2	7	20 000	92	222	314	718.49	3 015	151	3166	969.74
		10 000	250	133	383		4 469	82	4551	
		5 000	598	71	669		5 886	49	5935	
S3	5	20 000	142	249	391	221,48	3 016	200	3216	318.38
		10 000	465	161	626		4 521	121	4642	
		5 000	1 778	88	1866		8 604	72	8676	

Table 3. Experimental results

the best and worst initial states of the LFSR for all 3 systems. The CPU time needed for the analysis is presented as well.

For the first two systems S1 and S2 the cost of the procedure is determined only by the CPU time for the pseudorandom test pattern generation and by subsequent simulation of the test patterns for all cores in the system. For the third system S3 the CPU time includes also the time needed to generate the additional deterministic test patterns.

The full overview about the all possible hybrid BIST solutions for the three systems is presented in Figure 2 representing the memory cost as the function of the total test length. Based on these curves for an arbitrary memory constraint the corresponding total testing time can be found. The three constraints presented in Table 3 are also highlighted in Figure 2. It can be seen that the memory cost will increase very fast when reducing the length of the test sequence. It can be explained by the fact that in the beginning of the pseudorandom sequence nearly all test patterns are efficient, and nearly each pattern that is excluded from the pseudorandom part should be included into the deterministic part.

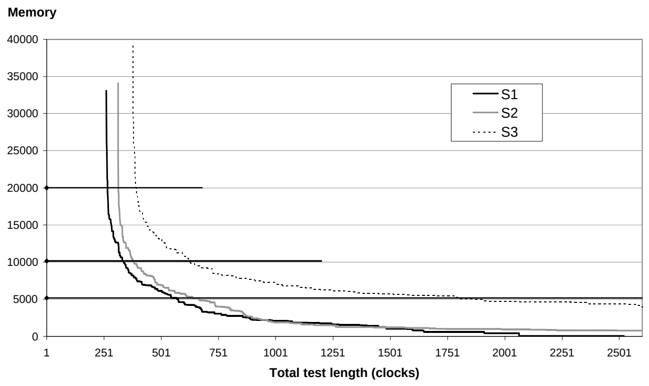

Figure 2. Memory usage as the function of the total test length for all three systems

A comparison of the curves of the memory cost as the function of the total test length for the best and for the worst initial pseudorandom sequences is depicted for the system S2 in Figure 3. This illustrates the importance of choosing best possible pseudorandom sequence for testing the system.

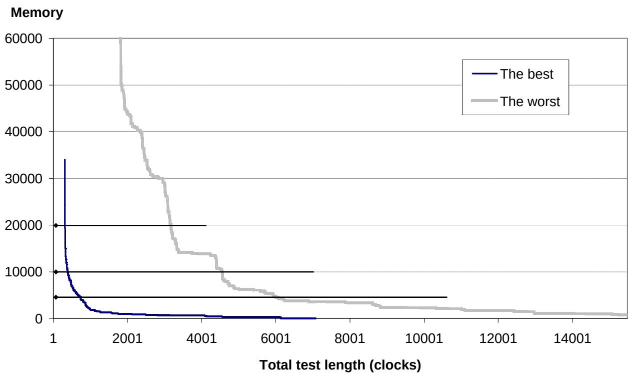

Figure 3. Memory usage as the function of the total test length for the best and the worst initial pseudorandom sequences

5. Conclusions

We have presented a new approach for the hybrid BIST in multi-core systems where the hybrid BIST idea is extended with the concept of test pattern broadcasting, where the deterministic test set of each core is applied in parallel to all other cores in a similar way as the pseudorandom test patterns. For this new architecture we have formulated the task to minimize the total test time of the hybrid BIST at given memory limitations for storing deterministic test patterns. We have proposed a straightforward algorithm for calculating a possible combination between pseudorandom and deterministic test sequences, where we neglect the optimal solutions for individual cores in favour of finding a near-optimal solution for the whole system. The described procedure doesn't guarantee minimal test length, however, the procedure is simple (similar to the greedy algorithm) and fast. The latter is demonstrated also by corresponding experimental results. We have also analyzed the impact of pseudorandom test quality for the overall test solution and the result was illustrated with the experimental results as well.

Although the current work covers only combinatorial circuits, it can easily be extended also for full-scan sequential circuits and can be considered as a future work.

The method proposed can be used first, as a cheap practical solution, and second, as a quickly computable reference for comparison with more sophisticated optimization algorithms to be developed in future.

References

[1] G. Jervan, Z. Peng, R. Ubar, "Test Cost Minimization for Hybrid BIST," *IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems (DFT'00)*, pp.283-291, Yamanashi, Japan, October 2000.

[2] G. Jervan, P. Eles, Z. Peng, R. Ubar, M. Jenihhin, "Test Time Minimization for Hybrid BIST of Core-Based Systems," *IEEE Asian Test Symposium 2003 (ATS'03)*, Xian, China, November 2003 (accepted for publication).

[3] Y. Zorian, “A distributed BIST control scheme for complex VLSI devices”, *Proceedings of the IEEE VLSI Test Symposium (VTS)*, pp. 4-9, Atlantic City, NJ, April 1993.

[4] R. Chou, K. Saluja, and V. Agrawal, Scheduling Tests for VLSI Systems Under Power Constraints, *IEEE Transactions on VLSI Systems*, Vol. 5, No. 2, pp. 175-185, June 1997.

[5] 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.

[6] K. Chakrabarty, “Test Scheduling for Core-Based Systems Using Mixed-Integer Linear Programming”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 19, No. 10, pp. 1163-1174, October 2000.

[7] V. Iyengar, K. Chakrabarty, E. J. Marinissen, ”Test Wrapper and Test Access Mechanism Co-Optimization for System-on-chip,” *Journal of Electronic Testing; Theory and Applications (JETTA)*, Vol. 18, no. 2, pp. 213-230, April 2002.

[8] E. Larsson, Z. Peng, “An Integrated Framework for the Design and Optimization of SOC Test Solutions”, *Journal of Electronic Testing; Theory and Applications (JETTA), for the Special Issue on Plug-and-Play Test Automation for System-on-a-Chip*, Vol. 18, no. 4/5, pp. 385-400, August 2002.

[9] G. Jervan, Z. Peng, R. Ubar, H. Kruus, “A Hybrid BIST Architecture and its Optimization for SoC Testing,” *IEEE 2002 3rd International Symposium on Quality Electronic Design (ISQED'02)*, pp. 273-279, San Jose, CA, March 2002.

[10] G. Jervan, P. Eles, Z. Peng, R. Ubar, M. Jenihhin, ”Hybrid BIST Time Minimization for Core-Based Systems with STUMPS Architecture,” *IEEE Int. Symposium on Defect and Fault Tolerance in VLSI Systems (DFTS'03)*, pp. 225-232, Cambridge, Mass, USA, November 2003.

[11] R. Ubar, M. Jenihhin, G. Jervan, Z. Peng, “Test Time Minimization for Hybrid BIST with Test Pattern Broadcasting,“ *21th Norchip Conference*, Riga, Latvia, November 2003 (accepted for publication).