

The Design and Optimization of SOC Test Solutions

Erik Larsson, Zebo Peng
Department of Computer Science,
Linköpings Universitet, Sweden
{erila, zebpe}@ida.liu.se

Gunnar Carlsson
CadLab Research Center
Ericsson
gunnar.carlsson@era.ericsson.se

Abstract¹

We propose an integrated technique for extensive optimization of the final test solution for System-on-Chip using Simulated Annealing. The produced results from the technique are a minimized test schedule fulfilling test conflicts under test power constraints and an optimized design of the test access mechanism. We have implemented the proposed algorithm and performed experiments with several benchmarks and industrial designs to show the usefulness and efficiency of our technique.

1 Introduction

The testing of System-on-Chip (SOC) is a crucial and time consuming problem due to the increasing design complexity. It is therefore important to provide the test designer with support to develop an efficient test solution and it is our belief that a designer would benefit from:

- for early design space exploration, an integrated design framework which deals with testability as well as performance and cost, and
- for the final solution, a combined technique for extensive optimization of the test schedule and the design and optimization of the test access mechanism (TAM) minimizing the test time and the routing of the TAM.

A design framework for fully BISTed systems, where each testable unit has its dedicated test source and test sink and no test conflicts exist, has been proposed by Benso *et al.* [1] and recently we have proposed, for general systems, an integrated framework where:

- tests are scheduled to minimize the test time,
- a TAM is designed and minimized,
- test sets for each testable unit are selected,
- test resources are floor-planned, and
- tests are parallelized (scan-chain division of scan-based cores).

The above set of tasks is performed in a single algorithm which considers test conflicts, power limitation and test resource constraints [2].

Our framework is suitable for early design space exploration due to its low computational complexity, which is an

advantage since it will be used iteratively many times. However, when the design exploration process is completed, an extensive optimization is to take place for the final solution in order to reach a near-optimal solution. Such optimization is only performed for the final solution when a near optimum is desired which justifies a higher computational cost.

Furthermore, we are interested in the efficiency of our previously proposed approach for large industrial designs. The test scheduling problem has been showed by Chakrabarty [3] to be an NP-complete problem that justifies the use of heuristics. Several techniques have been proposed [3,4,5,6,7]; however, all approaches have been evaluated using rather small benchmarks. For such benchmarks, a technique based on Mixed-Integer Linear Programming (MILP) has been proposed by Chakrabarty [3]. A disadvantage of such approach is the complexity of solving the MILP model. The size of the model grows quickly with the number of tests making it infeasible for large industrial designs.

The objective of this paper is to:

- show that our previously proposed technique is efficient for large industrial designs,
- evaluate the deviation of our previously proposed technique in respect to extensive optimization of:
 - test time and
 - combined cost of test time and TAM,
- evaluate other previously proposed techniques, and
- provide the test designer with a tool for the final optimization.

The objective is achieved by implementing a Simulated Annealing [8] algorithm which is used to perform extensive experiments on test scheduling and for integrated test scheduling and TAM design on several benchmarks and on an Ericsson design consisting of 170 tests.

The rest of the paper is organized as follows. An overview of related work is given in Section 2 and preliminaries are in Section 3. The algorithm for the TAM design is described in Section 4 and the algorithm for early design exploration is in Section 5. The Simulated Annealing algorithm is discussed in Section 6 and experimental results are presented in Section 7. The paper is concluded in Section 8.

1. This work has partially been supported by the Swedish Agency for Innovation Systems (VINNOVA) and Ericsson.

2 Related Work

2.1 Test Scheduling

The basic problem in test scheduling is to assign a start time for all tests while fulfilling all constraints. In order to minimize the test time it is important to schedule the tests as concurrently as possible. However, conflicts and limitations must be carefully considered. For instance, the tests may be in conflict with each other due to the sharing of test resources; and power consumption must be controlled, otherwise the system may be damaged during test.

A test is a set of test vectors produced or stored at a test source (placed on-chip or off-chip) and the test response from the test is evaluated at a test sink (placed on-chip or off-chip). If a testable unit is tested by several test sets (often an external test set and a on-chip test set are required to reach high test quality), a test conflict occur and only one test set can be applied at a time.

The power consumption during testing mode is usually higher compared to normal operation [9]. For instance, consider a memory, which is often organized in memory banks. During normal operation, only a single bank is activated. However, during testing mode, in order to test the system in the shortest possible time it is desirable to concurrently activate as many banks as possible [6].

Zorian proposed a technique for the testing of fully BISTed system [5] and for general systems an analytic approach is proposed by Chou *et al.* to minimize test time while considering test conflicts and test power [6]. Recently, Muresan *et al.* [7] proposed an approach where test time is minimized while test conflicts and test power consumption are considered and Chakrabarty [3,4] proposed an approach minimizing test time for core-based systems.

2.2 Test Access Mechanism

A test infrastructure is responsible for the transportation of test vectors from test sources to cores under test and test responses from cores under test to test sinks and it consists of two parts; one for the test data transportation and one for the control of the transportation.

In the test scheduling approach for fully BISTed systems proposed by Zorian[5], tests at blocks placed physically close to each other are grouped in the same test session which allows the same control structure to be used for all tests in the session which minimizes the routing of control wires. In general, systems are not tested with a BIST structure only for each testable unit and therefore a TAM is required. Several approaches have been proposed [10,11,12]. Chakrabarty proposed an integer linear programming (ILP) for the allocation of test bus width[13] and the effect on test time for systems using various design styles for test access with the TestShell wrapper has been analysed by Aertes *et al.*[14].

3 Preliminaries

An example of a system under test is given in Figure 1 where each core is placed in a wrapper such as TestShell [15] or P1500 [16] in order to achieve efficient test isolation and to ease test access. Each core consists of at least one block with added DFT technique and in this example all blocks are tested using the scan technique. The test access port (*tap*) is the connection to an external tester and the test resources, *test generator 1*, *test generator 2*, *test response evaluator 1* and *test response evaluator 2*, are implemented on the chip and the system can be modelled as a *design with test* [2, 18].

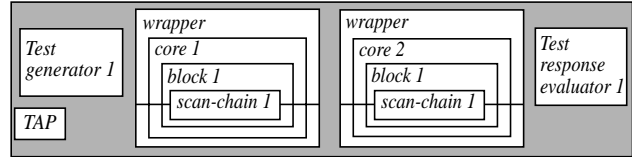


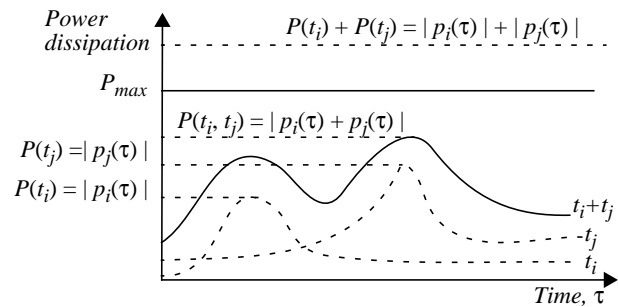
Figure 1. An example system.

3.1 Test Power Consumption

Generally speaking, there are more switching activities during the testing mode of a system compared to when it is operating under the normal mode. An example illustrating the test power dissipation variation over time τ for two test t_i and t_j is given in Figure 2. Let $p_i(\tau)$ and $p_j(\tau)$ be the instantaneous power dissipation of two compatible tests t_i and t_j , respectively, and $P(t_i)$ and $P(t_j)$ be the corresponding maximal power dissipation.

If $p_i(\tau) + p_j(\tau) < P_{max}$, the two tests can be scheduled at the same time. However, instantaneous power of each test vector is hard to obtain. To simplify the analysis, a fixed value $p_{test}(t_i)$ is usually assigned for all test vectors in a test t_i such that when the test is performed the power dissipation is no more then $p_{test}(t_i)$ at any moment.

The $p_{test}(t_i)$ can be assigned as the average power dissipation over all test vectors in t_i or as the maximum power dissipation over all test vectors in t_i . The former approach could be too optimistic; leading to an undesirable test schedule which exceeds the test power constraints. The latter could be too pessimistic; however, it guarantees that the



$p_i(\tau)$ = instantaneous power dissipation of test t_i

$P(t_i) = |p_i(\tau)|$ = maximum power dissipation of test t_i

Figure 2. Power dissipation as a function of time [6].

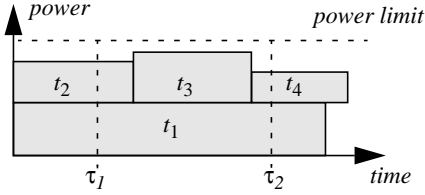


Figure 3. Example of test scheduling.

power dissipation will satisfy the constraint [6]. Usually, in a test environment the difference between the average and the maximal power dissipation for each test is often small since the objective is to maximize the circuit activity so that it can be tested in the shortest possible time [6]. Therefore, the definition of power dissipation $p_{test}(t_i)$ for a test t_i is usually assigned to the maximal test power dissipation ($P(t_i)$) when test t_i alone is applied to the device. This simplification was introduced by Chou *et al.* [6] and has been used by Zorian [5] and by Muresan *et al.* [7]. We will use this assumption also in our approach. In this paper, an additive model used by Zorian [5], Chou *et al.* [6] and Muresan *et al.* [7] for power consumption is assumed. The maximal power consumption should not exceed the power constraint, p_{max} , for a schedule to be accepted. That is, $p_{sch}(0, \infty) \leq p_{max}$.

3.2 Test Scheduling

Given a system as in Figure 1 where core 1 is tested by t_1 and core 2 by t_2, t_3 and t_4 , the test scheduling problem can be seen as placing all tests in a diagram as in Figure 3 while satisfying all constraints. The basic difference between our scheduling technique [2,18] and the approaches by Zorian [5] and Chou *et al.* [6], besides that we design the TAM while scheduling the tests, is that Zorian and Chou *et al.* do not allow new tests to start until all tests in a session are completed. It means that t_3 and t_4 are not allowed to be scheduled as in Figure 3. In the approach proposed by Muresan *et al.* [7], t_3 is allowed to be scheduled if it is completed no later than t_1 (Figure 3), however, t_4 is not allowed to start before t_1 finishes.

In our approach it is optional if tests may start before all tests in a session are completed or not. If it is allowed, t_3 and t_4 can be scheduled as in Figure 3, which gives more flexibility.

4 Test Access Mechanism Design

When a designer is about to design the TAM, two major problems must be solved, namely:

- the design and the routing of the TAM, and
- the scheduling of the tests on the TAM.

In order to minimize the routing, few and short wires are desired. However, such approach may increase the test time of the system. For instance, consider System S [4] (Table 1) where we added the floor planning (x,y co-ordinates for each core). The BIST tests require 1 clock cycle while the external tests are ten times slower.

Core	Index i	External test cycles, e_i	BIST cycles, b_i	Placement	
				x	y
c880	1	377	4096	10	10
c2670	2	15958	64000	20	10
c7552	3	8448	64000	10	30
s953	4	28959	217140	20	30
s5378	5	60698	389214	30	30
s1196	6	778	135200	30	10

Table 1. Test data for the cores in System S.

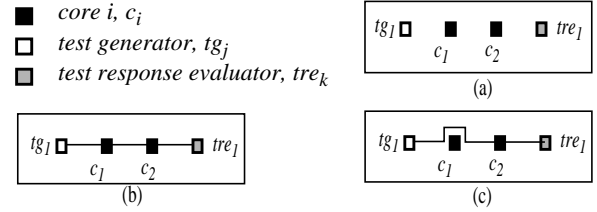


Figure 4. TAM design alternatives of example (Fig.1).

A minimal TAM would be a single wire starting at the TAP, connecting all cores and ending at the TAP. However, such TAM design would require all tests to be scheduled in a sequence leading to long test time.

The system (for instance the example system in Figure 1 or System S) can be modelled as a directed graph, $G=(V,A)$, where V consists of the set of blocks (the testable units), B , the set of test sources, R_{source} , and the set of test sinks, R_{sink} , *i.e.* $V=B \cup R_{source} \cup R_{sink}$ [2,18]. An arc $a_i \in A$ between two vertices v_i and v_j indicates a TAM (a wire) where it is possible to transport test data from v_i to v_j . Initially no TAM exists in the system, *i.e.* $A=\emptyset$. However, if the functional infrastructure may be used, it can be included in A initially.

When adding a TAM between a test source and a core or between a core and a test sink, and the test data has to pass another core, c_i , several routing options are possible:

1. through the core c_i using the transparent mode of the core;
2. through an optional bypass structure of core c_i ; and
3. around core c_i where the TAM is not connected to the core.

The model in Figure 4(a) of the example system in Figure 1 illustrates the alternatives 1 and 2 (Figure 4 (b)) and alternative 3 (Figure 4 (c)). In alternatives 1 and 2 the same TAM can be used when testing c_1 and c_2 . However, a delay may be introduced when the core is in transparent mode or its by-pass structure is used as in the TestShell proposed by Marinissen *et al.* [15]. On the other hand, Marinissen *et al.* recently proposed a library of wrapper cells allowing a flexible design [17] where it is possible to design non-clocked bypass structures of TAM width. In the following, we assume that bypass may be solved by a non-delay

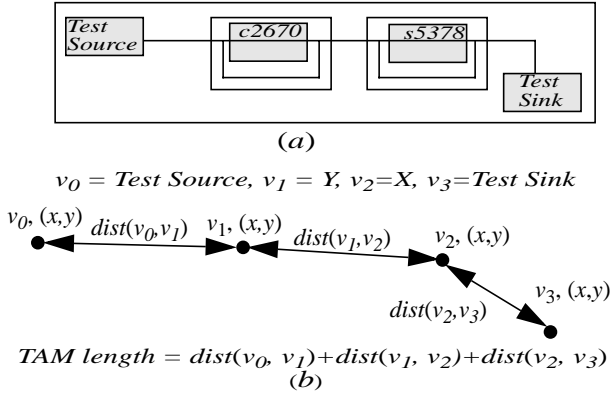


Figure 5. Computing the TAM length.

mechanism or that the delay due to clocked by-pass is negligible.

A test wire, w_i , is a path of edges $\{(v_0, v_1), \dots, (v_{n-1}, v_n)\}$ where $v_0 \in R_{\text{source}}$ and $v_n \in R_{\text{sink}}$. Let Δy_{ij} be defined as $|y(v_i) - y(v_j)|$ and Δx_{ij} as $|x(v_i) - x(v_j)|$, where $x(v_i)$ and $y(v_i)$ are the x -placement respectively the y -placement for a vertex v_i and the distance between vertex v_i and vertex v_j is given by:

$$\text{dist}(v_i, v_j) = \sqrt{(\Delta y_{ij})^2 + (\Delta x_{ij})^2}. \quad 1$$

The total length of a path is the sum of all individual edges. An example to illustrate the calculation of the length of a test wire defined as a path is in Figure 5.

5 The Design Space Exploration Algorithm

The algorithm for test scheduling and TAM design defined within the integrated test framework is used for design space exploration [2, 18]. It initially sorts the tests according to a key k which characterizes $\text{test-power}(p)$, $\text{test-time}(t)$ or $\text{test-power} \times \text{test-time}(p \times t)$. The algorithm can basically be divided into four parts:

- constraint checking,
- test resource placement,
- TAM design and routing, and
- test scheduling.

A main loop is terminated when there are selected and scheduled test for all testable units in the system and a designed TAM. In each iteration over the tests, a test is checked if it at the moment satisfy all constraints. If so, an existing TAM is selected or if no TAM is available, a new is designed and the test is scheduled.

An example of the produced results from the algorithm using System S [4] (Table 1) are the TAM design as in Figure 6 and the test schedule as in Figure 7. The TAM buses 1 to 5 in Figure 6 correspond to the TAM 1 to 5 in Figure 7. For instance, b_5 is the BIST test of core indexed 5 (s5378) and e_5 is the external test of s5378 (note that the

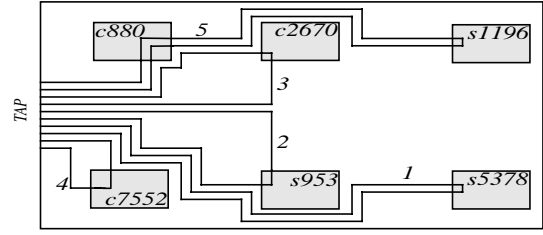


Figure 6. TAM design using our heuristic on System S.

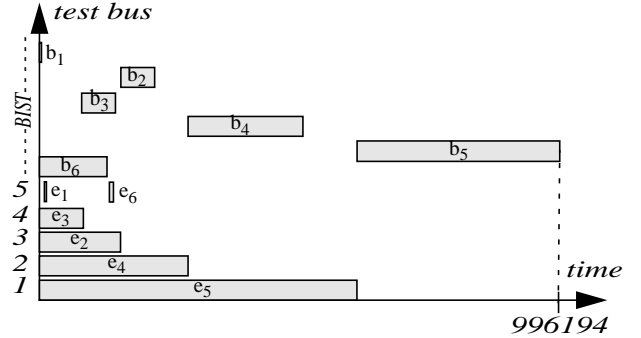


Figure 7. Test bus schedule on System S using our heuristic.

BIST tests such as b_5 do not require a TAM and they are shown in the part marked as BIST in Figure 7).

The computational complexity for the algorithm, where the TAM design is excluded in order to make it comparable with other approaches, comes mainly from the initial sorting of the tests and the two loops, *i.e.* a worst case complexity of $O(n^2)$. The approach by Chakrabarty [3] has a worst case complexity of $O(n^3)$

6 Simulated Annealing

We outline the Simulated Annealing (SA) technique and describe how it is adopted to be used for scheduling and TAM design. The SA technique proposed by Kirkpatrick *et al.* [8] uses a hill-climbing mechanism to avoid getting stuck at local optimum.

6.1 The Simulated Annealing Algorithm

The SA algorithm (Figure 8) starts with an initial solution and a minor modification of it creates a neighbouring solution. The cost of the new solution is evaluated and if the new solution is better than the previous, the new solution is kept. A worse solution can be accepted at a certain probability, which is controlled by a parameter referred to as temperature.

The temperature is decreased during the optimization process, and the probability of accepting a worse solution decreases with the reduction of the temperature value and when the temperature value is approximately zero, the optimization terminates.

- 1: Construct initial solution, x^{now} ;
- 2: Initial Temperature: $T:=TI$;
- 3: while stop criteria not met do begin
- 4: for $i = 1$ to TL do begin
- 5: Generate randomly a neighboring solution
 $x' \in N(x^{now})$;
- 6: Compute change of cost function
 $\Delta C := C(x') - C(x^{now})$;
- 7: if $\Delta C \leq 0$ then $x^{now} = x'$;
- 8: else begin
- 9: Generate $q := \text{random}(0, 1)$;
- 10: if $q < e^{-\Delta C/T}$ then $x^{now} = x'$;
- 11: end;
- 12: end;
- 13: Set new temperature $T := \alpha \times T$;
- 14: end;
- 15: Return solution corresponding to the minimum cost function;

Figure 8. Simulated Annealing algorithm.

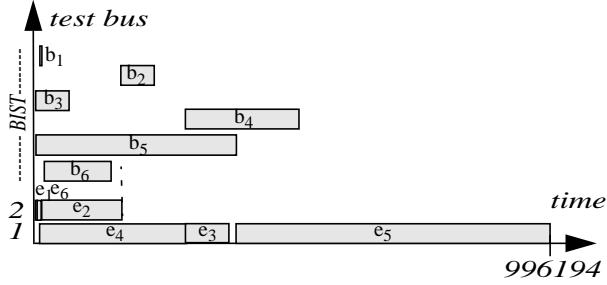


Figure 9. Test schedule on System S using SA.

6.2 Initial Solution and Parameter Selection

We use the algorithm with an initial sorting of the tests based on p (using t and $p \times t$ results after optimization in the same cost) within the integrated test framework (Section 5) to create the initial solution [2,18]. An example of an initial solution produced on System S is in Figure 6 and Figure 7.

The parameters, initial temperature TI , the temperature length TL and temperature reduction factor α ($0 < \alpha < 1$) are determined based on experiments.

6.3 Neighbouring Solution in Test Scheduling

In the case when only test scheduling is considered, *i.e.* the TAM is not considered or it is fixed and can be seen as a resource, we create a neighbouring solution by randomly selecting a test from an existing schedule and schedule it as soon as possible but not at the same place as it was in the original schedule.

For instance, creating a neighboring solution given a test schedule as in Figure 3, we randomly select a test, let say t_2 . We try to schedule t_2 as soon as possible but not with the same starting time as it had while fulfilling all constraints. Test t_2 was scheduled to start at time 0 and no new starting point exists where constraints are fulfilled until end of t_1 where t_2 is scheduled. In this case, the test time increases after the modification (getting out of a possible local minimum), however, only temporarily since in the next iteration a test may be scheduled at time 0 (where t_2 used to be).

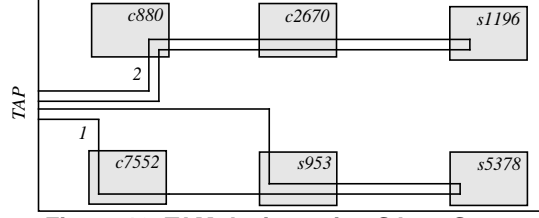


Figure 10. TAM design using SA on System S.

6.4 Neighbouring Solution in Test Scheduling and TAM Design

When both the test time and the TAM design are to be minimized a neighbouring solution is created by randomly adding or deleting a wire and then the tests are scheduled on the modified TAM.

If the random choice is to add a wire, a test is randomly selected and a wire is added from its required test source to the core where the test is applied and from the core to the test sink for the test. For instance, if e_3 in System S (Table 1) is selected, a wire is added from the TAP to core $c7552$ and from core $c7552$ to the TAP. If the random choice is to delete a wire, a similar approach is applied. However, a check is performed to make sure that all tests can be applied. After the TAM modification, all tests are re-scheduled.

6.5 Cost function

The cost function of a test schedule, S , and the TAM, A , is:

$$C(S, A) = \beta_1 \times T(S) + \beta_2 \times L(A) \quad 2$$

where: $T(S)$ is the test application time for a sequence of tests, S , $L(A)$ is the total length of the TAM, β_1, β_2 are two designer-specified constants used to determine the importance of the test time and the test bus.

The test application time, $T(S)$, for a schedule, S , is:

$$T(S) = \{\tau_{end}(t_i) \mid \forall t_i(\max\{\tau_{end}(t_i)\}), t_i \in S\} \quad 3$$

and the length, $L(A)$, of the TAM, A , is given by:

$$\sum_{w_j \in A} \sum_{j=0}^{|w_i|-1} dist(v_j, v_{j+1}), v_j, v_{j+1} \in w_i \quad 4$$

For the test schedule, S , produced by SA for System S (Figure 9) the test time, $T(S)$ is 996194 (the end time of test e_5) and the length of the TAM (Figure 10) is 160. Note the test time is optimal in this case since the two tests, b_5 and e_5 , for core $s5378$, determines the total test time. Comparing this to the results produced by our heuristic [2] shows that test time is the same while the TAM is reduced from 320 (Figure 6) to 160 (Figure 10).

7 Experimental Results

We have used the System S [4] which has test conflicts (Table 1) while all other benchmarks and designs have constraints on tests and power like the benchmark presented by Muresan *et al.* [7].

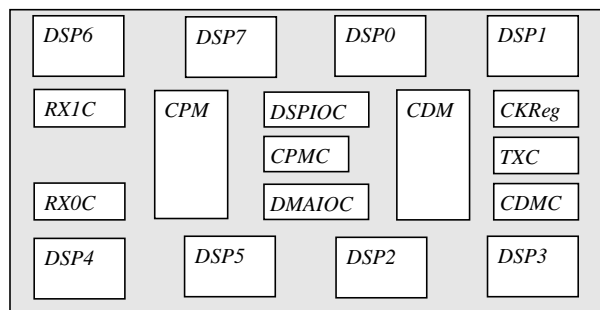


Figure 11. The Ericsson design.

We have used two industrial designs, the ASIC Z presented by Zorian [5] and with data added by Chou *et al.* [6] and placement (x, y) co-ordinates [2]. The design is fully BISTed and the maximal allowed power dissipation is 900 mW. The maximal power dissipation for the Ericsson design [18] is 5125 mW and it consists of 8 DSP cores plus additional logic cores and memory banks, as illustrated in Figure 11 and design characteristics as in Table 2 where the following notations are used:

- n : DSP core ($0 \leq n \leq 7$),
- i : common program memory (CPM) bank ($0 \leq i \leq 7$),
- j : common data memory (CDM) bank ($0 \leq j \leq 9$),
- l : local data memory (LDM) bank at a DSP core ($0 \leq l \leq 3$),
- m : local memory (LZM) bank at a DSP core ($0 \leq m \leq 1$).

All logic blocks in the Ericsson design are tested by two test sets, one using external test resources and the other using on-chip resources while memories are tested with one test set. It results in a total of 170 tests in the design. The test access port may be used by more than one test concurrently. However, the other test resources may not be used concurrently. Furthermore, only one test set may be applied concurrently to each block.

For the experiments, we allow tests to start as soon as possible, for the cost function (Section 6.5) $\beta_1 = \beta_2 = 1$ unless stated and we have used a Sun Ultra Sparc 10, 450 MHz CPU, 256 MB RAM.

7.1 Test Scheduling

The results from the experiments on design Muresan [7] is in the first group of Table 3. The test time using the approach by Muresan *et al.* is 29 time units and the results using our approach with initial sorting of tests based on *test power* (p), *test time* (t) and *test power* \times *test time* ($p \times t$) are 28, 28 and 26, respectively, all produced within a second. Our SA (TI=400, TL=400, $\alpha=0.97$) improves to 25 time units using 90 sec.

When idle power is not considered on ASIC Z, the test schedules using our approach with the initial sorting of tests based on p , t and $p \times t$ (second group in Table 3) all result in a test application time of 262. The SA was running for 74

Block	Test number	Test time	Test Power	Test source	Test sink	
RX0C	1	970	375	TAP	TAP	
	2	970	375	TG ₀	TRA ₀	
RX1C	3	970	375	TAP	TAP	
	4	970	375	TG ₀	TRA ₀	
DSPIOC	5	1592	710	TAP	TAP	
	6	1592	710	TG ₀	TRA ₀	
CPMC	7	480	172	TAP	TAP	
	8	480	172	TG ₀	TRA ₀	
DMAIOC	9	3325	207	TAP	TAP	
	10	3325	207	TG ₀	TRA ₀	
CKReg	11	505	118	TAP	TAP	
	12	505	118	TG ₀	TRA ₀	
CDMC	13	224	86	TAP	TAP	
	14	224	86	TG ₀	TRA ₀	
TXC	15	364	140	TAP	TAP	
	16	364	140	TG ₀	TRA ₀	
CPM _i	17+i	239	80	TG ₁	TRA ₁	
	CDM _j	25+j	369	64	TG ₁	TRA ₁
DSP ⁿ	LPM	35+17×n	46	16	TG _{n,0}	TRA _{n,0}
	LDM _l	36+17×n+1	92	8	TG _{n,0}	TRA _{n,0}
	LZM _m	40+17×n+m	23	2	TG _{n,0}	TRA _{n,0}
	Logic ₀	17×n+42	4435	152	TAP	TAP
	Logic ₁	17×n+43	4435	152	TG _{n,1}	TRA _{n,1}
		17×n+44	4435	152	TAP	TAP
	Logic ₂	17×n+45	4435	152	TG _{n,1}	TRA _{n,1}
		17×n+46	7009	230	TAP	TAP
	Logic ₃	17×n+47	7009	230	TG _{n,1}	TRA _{n,1}
		17×n+48	7224	250	TAP	TAP
	Logic ₄	17×n+49	7224	250	TG _{n,1}	TRA _{n,1}
		17×n+50	7796	270	TAP	TAP
	17×n+51	7796	270	TG _{n,1}	TRA _{n,1}	

Table 2. The Ericsson design characteristics.

seconds (TI=400, TL=400 and $\alpha=0.97$) and found a solution at a cost of 262, *i.e.* no improvement.

In the experiments considering idle power (third group of Table 3), our heuristic approach with an initial sorting based on p , t and $p \times t$ resulted in a solution of 300, 290 and 290, respectively, each produced within 1 second. The SA (TI=400, TL=400 and $\alpha=0.99$) produced a solution of 274 requiring 223 sec., *i.e.* a cost improvement in the range of 6% to 10%.

The results (group 4 in Table 3) produced by our heuristic after 1 second on Extended ASIC Z when not considering idle power are 313, 287 and 287 (initial sorting based on

Design	Approach	Test time	Diff. to SA	CPU
Muresan	SA	25	-	90 sec.
	Muresan [7]	29	16%	-
	test power [2]	28	12%	1 sec.
	test time [2]	28	12%	1 sec.
	test power \times test time[2]	26	4%	1 sec.
ASIC Z (1)	SA	262	-	74 sec.
	test power	262	0%	1 sec.
	test time	262	0%	1 sec.
	test power \times test time	262	0%	1 sec.
ASIC Z (2)	SA	274	-	223 sec.
	test power [2]	300	10%	1 sec.
	test time [2]	290	6%	1 sec.
Extended ASIC Z (3)	test power \times test time[2]	290	6%	1 sec.
	SA	264	-	132 sec.
	test power	313	18%	1 sec.
	test time	287	9%	1 sec.
Ericsson	test power \times test time	287	9%	1 sec.
	SA	30899	-	3260 sec.
	test power	37336	20%	3 sec.
	test time	34762	12%	3 sec.
	test power \times test time	34762	12%	3 sec.

Table 3. Test scheduling results.

p , t and $p \times t$). The SA optimization (TI=TL=400, $\alpha=0.97$) produces a solution at a cost of 264 running for 132 seconds, *i.e* a cost improvement in the range of 9% to 18%.

The results on the Ericsson design (fifth group of Table 3) are 37226, 34762, 34762 produced by our heuristic (within 3 sec.) with sorting based on t and $p \times t$. The SA algorithm (TI=200, TL=200, $\alpha=0.95$) produced a solution at 30899 after 3260 seconds.

7.2 Test Schedule and TAM Design

The test schedule (Figure 7) and TAM design (Figure 6) achieved using our heuristic for System S (Table 1) have a test time of 996194 and a TAM length of 320 computed within 1 second. The SA (TI=TL=100, $\alpha=0.99$) was running for 1004 seconds producing a test schedule (Figure 9) and a TAM design (Figure 10) with a test time of 996194 and a TAM length of 160, a TAM improvement of 50%.

ASIC Z is fully BISTed; however, here we assume all tests are applied using an external tester capable of supporting

Approach	SA	Test power	Test time	Test power \times test time
Test time	334	300	290	290
Diff to SA	-	-10%	-13%	-13%
TAM cost	180	360	360	360
Diff to SA	-	100%	100%	100%
Total Cost	514	660	650	650
Diff to SA	-	28%	21%	21%
Comp. cost	855 sec.	1 sec.	1 sec.	1 sec.
Diff to SA	-	-85400%	-85400%	-85400%

Table 4. TAM and scheduling results on ASIC Z.

ing several tests concurrently. All results are collected in Table 4 where, for instance, our heuristic produces a solution with a total cost of 650 (a test time of 290 and a TAM cost of 360) after 1 second. The SA (TI=TL=300, $\alpha=0.97$) produced after 855 seconds a solution at a cost of 514 (334 for test time and 180 for TAM). The test time results (Table 4) are in the range of 10% to 13% better using our fast heuristic (in all cases) compared to the SA optimization. However, the TAM results are much worse and the total cost improvements by the SA are in the range from 21% to 28%.

The results from experiments on Extended ASIC Z are in Table 5. Our heuristic approach with an initial sorting of the tests based on p produces a solution after 1 second with a test time of 313 and a TAM cost of 720, resulting in a total cost of 1033. The solution produced after 4549 seconds by our SA (TI=TL=200, $\alpha=0.97$) optimization has a test time of 270 and a TAM cost of 560. In this experiment, SA produced a better total cost (range 14% to 24%) as well as better cost regarding test time (range 6% to 16%) and TAM cost (range 18% to 29%).

The results on the Ericsson design are collected in Table 6. For instance, our heuristic with an initial sorting of the tests based on p results in a solution with a test time of 37336 and a TAM cost of 8245, which took 81 seconds to

Approach	SA	Test power	Test time	Test power \times test time
Test time	270	313	287	287
Diff to SA	-	16%	6%	6%
TAM cost	560	720	660	660
Diff to SA	-	29%	18%	18%
Total Cost	830	1033	947	947
Diff to SA	-	24%	14%	14%
Comp. cost	4549 sec.	1 sec.	1 sec.	1 sec.
Diff to SA	-	-454800%	-454800%	-454800%

Table 5. TAM and scheduling results on Extended ASIC Z.

produce. The total cost is 53826 when using $\beta_1=1$ and $\beta_2=2$. The SA (TI=TL=200, $\alpha=0.95$) optimization produced a solution with a test application time of 33082 and a TAM cost of 6910 after 15 hours. In all cases, the SA produces better results. Regarding test time the SA improvement is in the range 5% to 11%, for the TAM cost the in the range from 19% to 35% and the total cost in the range 10% to 15%.

For all experiments with the SA, the computational cost is extremely higher compared to our heuristics. A finer tuning of the SA parameters could reduce it, however, such extensive optimization is only used for the final design and therefore a high computational cost can be accepted.

Approach	SA	Test power	Test time	Test power \times test time
Test time	33082	37336	34762	34762
Diff to SA	-	11%	5%	5%
Test bus	6910	8245	9350	8520
Diff to SA	-	19%	35%	23%
Total Cost	46902	53826	53462	51802
Diff to SA	-	15%	14%	10%
Comp. cost	15h	81 sec.	79 sec.	62 sec.
Diff to SA		-66567%	-68254%	-86996%

Table 6. TAM and scheduling results on the Ericsson design.

8 Conclusions

For complex systems such as SOCs, it is a difficult problem to develop an efficient test solution due to the large number of factors involved. The workflow for a test designer consists of two consecutive parts: an early design space exploration and an extensive optimization for the final solution.

The latter is the focus of this paper where we have proposed and implemented a technique using Simulated Annealing for integrated test scheduling and TAM design. Our approach minimizes the test time as well as the TAM design while scheduling the tests and satisfying all test conflicts and power constraints.

We have used benchmarks and industrial designs to show the efficiency and usefulness of our approach. The experimental results shows that our previously proposed heuristic and the optimization using Simulated Annealing are able to handle industrial designs. Furthermore, our heuristic produces results at a very low computational cost, which are further improved by our Simulated Annealing algorithm.

References

[1] A. Benso et al., A High-Level EDA Environment for the Automatic Insertion of HD-BIST Structures, *Journal of Electronic Testing: Theory and Applications (JETTA)*, Vol. 16, No. 3, pp. 179-184, June 2000.

[2] E. Larsson, Z. Peng, An Integrated System-On-Chip Test Framework, *Proceedings of Design, Automation and Test in Europe*, pp. 138-144, Munchen, Germany, March 2001.

[3] K. Chakrabarty, Test Scheduling for Core-Based Systems Using Mixed-Integer Linear Programming, *Transactions on CAD of Integrated Circuits & Systems*, Vol. 19, No. 10, pp. 1163-1174, October 2000.

[4] K. Chakrabarty, Test Scheduling for Core-Based Systems, *Proceedings of ICCAD*, pp. 391-394, Nov. 7-11, 1999.

[5] Y. Zorian, A distributed BIST control scheme for complex VLSI devices, *Proceedings of VLSI Test Symposium.*, pp. 4-9, April 1993.

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

[7] V. Muresan et al, A Comparison of Classical Scheduling Approaches in Power-Constrained Block-Test Scheduling, *Proceedings of International Test Conference*, pp. 882-891, Atlantic City, NJ, October 2000.

[8] S. Kirkpatrick et al., Optimization by Simulated Annealing, *Science*, Vol. 220, No. 4598, pp. 671-680, 1983.

[9] S. Gerstendörfer and H.-J. Wunderlich, Minimized Power Consumption for Scan-Based BIST, *Proceedings of International Test Conference*, pp 77-84, Atlantic City, NJ, September 1999.

[10] M. Nourani and C. Papachristou, An ILP Formulation to Optimize Test Access Mechanisms in System-On-A-Chip Testing, *Proceedings of International Test Conference*, pp 902-910, Atlantic City, NJ, October 2000.

[11] K. Chakrabarty, Optimal Test Access Architectures for System-On-A-Chip, *ACM Transactions on Design Automation of Electronic Systems*, vol. 6, pp. 26-49, January 2001.

[12] K. Chakrabarty, Design of System-on-a-Chip Test Access Architectures Using Integer Linear Programming, *Proceedings of VLSI Test Symposium*, pp. 127-134, Montreal, Canada, April 2000.

[13] K. Chakrabarty, Design of System-on-a-Chip Test Access Architecture under Place-and-Route and Power Constraints, *Proceedings of Design Automation Conference*, pp. 432-437, Los Angeles, CA, June 2000.

[14] J. Aerts and E. J. Marinissen, Scan Chain Design for Test Time Reduction in Core-Based ICs, *Proceedings of International Test Conference*, pp. 448-457, Washington, DC, October 1998.

[15] E. J. Marinissen et al., A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores, *Proceedings of International Test Conference*, pp. 284-293, Washington, DC, October 1998.

[16] E. J. Marinissen et al., Towards a Standard for Embedded Core Test: An Example, *Proceedings of International Test Conference*, pp. 616-627, Atlantic City, NJ, Sep. 1999.

[17] E. J. Marinissen et al., Wrapper Design for Embedded Core Test, *Proceedings of International Test Conference*, pp. 911-920, Atlantic City, NJ, October, 2000.

[18] E. Larsson, An Integrated System-Level Design for Testability Methodology, *Ph. D. thesis 660*, Linköpings universitet, Sweden 2000.