Test Resource Partitioning and Optimization for SOC Designs

Erik Larsson[+]* and Hideo Fujiwara*

Embedded Systems Laboratory[+]
Linkopings Universitet
SE-582 83 Linkoping, Sweden
erila@ida.liu.se

Graduate School of Information Science[*]
Nara Institute of Science and Technology,
8916-5 Takayama, Ikoma, Nara 630-0101, Japan
fujiwara@is.aist-nara.ac.jp

Abstract[1]

We propose a test resource partitioning and optimization technique for core-based designs. Our technique includes test set selection and test resource floor-planning with the aim of minimizing the total test application time and the routing of the added TAM (test access mechanism) wires. A feature of our approach is that it pinpoints bottlenecks that are likely to limit the test solution, which is important in the iterative test solution development process. We demonstrate the usefulness of the technique through a comparison with a test scheduling and TAM design tool.

1 Introduction

Developing an efficient test solution for an SOC (System-on-Chip) design is a complicated task due to the wide range of design options and the high number of issues to be optimized. A typical SOC design consists of a set of connected pre-designed cores and dedicated UDL (user-defined logic) blocks. Each testable unit (core, UDL block and interconnection) has its test method and the task when developing a test solution is to:

- perform test set selection for each testable unit,
- floor-plan the test resources (test sources and test sinks) corresponding to the selected test sets,
- design the TAM (test access mechanism) to connect the selected test resources and the testable units design, and
- determine the order of the tests, *i.e.* schedule the test

with the objective to minimize the test application time and the routing of TAM wires (added for test data transportation) while considering test conflicts and power constraints.

The four items, *test set selection*, *test resource floor-planning*, *TAM design* and *test scheduling*, are highly inter-dependent. The test time can be minimized by scheduling the tests as concurrent as possible, however, the possibility of concurrent testing depends on the size of the TAM connecting the test resources. The placement of the test resources has a direct impact on the length of the TAM wires. Finally, the way the test sets for each testable unit are partitioned over the test resources impacts the TAM design.

We have proposed an integrated technique for test scheduling and TAM design minimizing both the test application time and the TAM design while considering test conflicts and power consumption [7,9]. In this paper, we address the creation and optimization of the TRS (*test resource specification*). The objective of our work is to create a TRS that together with the design specification will be the inputs to a test scheduling and TAM design tool and hence result in an efficient test solution, *i.e.* minimal test application time and minimal routing of the TAM wires. We analyze the complexity of the design space and we propose an estimation-based TRS creation technique. We also address the refinement of a test specification by proposing an iterative approach where design bottlenecks are detected using Gantt charts [2].

The rest of the paper is organized as follows. In Section 2 we review previous work and preliminaries such as system modelling are described in Section 3. Our technique for test resource partitioning and optimization is presented in Section 4. The experimental results are in Section 5 and finally conclusions are in Section 6.

2 Related Work

Several approaches addressing issues that are to be considered when developing a SOC test solution have been proposed. Zorian proposed a scheduling technique that minimizes the test time while considering test power consumption [15]. The technique assumes that each testable unit has its pre-determined and dedicated BIST (Built-In Self-Test) resource. Iyengar *et al.* investigated the use of preemptive test scheduling, which means that the pre-determined tests can be interrupted and resumed later. It actually means that the test sets are partitioned on-the-fly into several test sets. Sugihara *et al.* investigated the partitioning of test sets into on-chip test (BIST) and off-chip test using an ATE (Automatic Test Equipment) [14]. Jervan *et al*. also investigated test set partitioning by making use of processor cores and memories for test vector generation and storage [6]. Arabi proposed a technique to reduce test time by test identical cores simultaneously [1]. The power consumption is becoming a problem since exceeding the power limitation may damage the system. Saxena *et al.* proposed a gating scheme reducing the test power

1. This work has been supported by the Japan Society for the Promotion of Science (JSPS) under grant P01735 and the Swedish National Program on Socware.

consumed during the shift process [13]. Hetherington *et al*. discussed several important test limitations such as ATE bandwidth and memory limitations [4]. All the addressed problems are each important. However, it is also important to consider them all from a system test perspective.

3 SOC Test Model

A system consisting of three testable units, core A, core B and a UDL block will serve as an example system (Figure 1). Each testable unit is tested by applying at least one set of test vectors (test stimuli) where each test set is stored or generated at a *test source* and the test response is stored or analyzed at a *test sink*. Each testable unit may have its dedicated test source and test sink or may share either test source or test sink with other testable units or may share both test source and test sink. The *test resources* (test source and test sink) can be placed on-chip or off-chip where an ATE is a typical off-chip test resource and an LFSR (Linear-Feedback Shift-Registers) is a typical on-chip test source. In general, any combination of test resources is possible. The test stimuli at a test source can be generated off-chip and analyzed on-chip. It is also possible to store the test stimuli in an on-chip memory making the test source on-chip and storing the test response in a test sink placed off-chip.

A testable unit can also be tested by several test sets. One test set can be for stuck-at fault testing, one for at-speed testing and one for functional testing. Furthermore, it is also possible to test a testable unit with one test set stored at an ATE, one test set generated by an LFSR, and one test set stored in a memory. The test solution is highly dependent on the partitioning of the test sets for the testable units. A limited test set at an ATE often results in the same fault coverage as a larger test set generated by an LFSR. However, an LFSR, if it is dedicated to a testable unit, does not require a TAM, *i.e.* the routing is minimized.

Executing a test at a testable unit means that the test stimuli is transported from the required test source to the testable unit and the test response is transported from the testable unit to the required test sink. The test data (test stimuli and test response) is transported using TAM wires. A *test wrapper* is the interface between the testable unit and the TAM added in order to ease test access. A testable unit with a test wrapper is *wrapped* while a testable unit with no wrapper is *unwrapped* [11]. For instance in Figure 1, core A is place in a wrapper making it wrapped while the UDL block has no wrapper making it unwrapped.

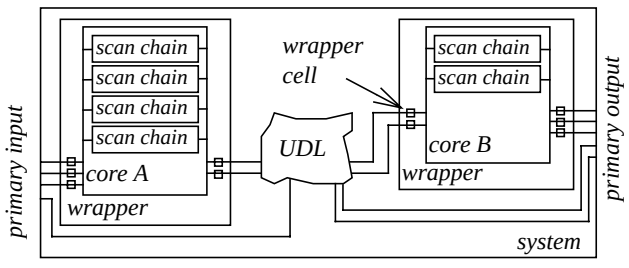

Figure 1. Example system.

The testing of a wrapped testable unit is different from the testing of an unwrapped. The testing of the wrapped core A is performed by placing the wrapper in *internal test mode* and test stimuli is transported from the required test source using a set of TAM wires to the core and the test response is transported from the core using a set of TAM wires to the test sink. In the case of an unwrapped testable unit such as the UDL block (Figure 1), the wrappers at core A and B are placed in *external test mode*. The test stimuli is transported from the required test source on the TAM via core A to the UDL block and the test response is transported via core B to the TAM and to the test sink.

A wrapper can at any time be in one of the three modes; *internal mode*, *external mode* and *normal operation mode*. It means that testing of core A and the UDL block cannot be performed at the same time due to the wrapper, there is a test conflict. A test conflict also occurs when a testable unit is to be tested by several test sets since it is only possible to apply one test set at each testable unit at a time. And finally, a test conflict may occur when test resources are shared.

The *input specification* to a test design tool consists of two parts, the *design specification* and the *test resource specification*. The SOC test integrator is given a design specification and the task is to design a test schedule and design the TAM while minimizing the test application time and the TAM routing while considering test conflicts and test power constraints. The design specification is determined, however, the test resource specification is to be determined by the SOC test integrator. The problem is for a given design specification to develop a test resource specification, which together with the design specification will result in an efficient test solution. The development of a test resource specification include:

- the test set selection for each testable unit,
- the placement of each test source, and
- the placement of each test sink.

The number of design options when developing the test resource specification can be high. Assume that each block (testable unit) $b_{ij} \in B$ at a core c_i has $|T_{ij}|$ possible combinations of test sets where each test set can be placed at n_{ij} positions and each test set can be modified in m_{ij} ways where a high number of TAM wires reduces the test time and vice versa. The number of possibilities are:

$$\prod_{i=1,\, j=1}^{|B|} |T_{ij}| \times n_{ij} \times m_{ij}$$

For a small system consisting of only two cores; each with two test sets where each test set have two possible placements and each test set can be modified in two ways, the number of design alternatives are: $(2\times2\times2)^2=64$.

We will use our proposed system model [7,9], which we partition into a design specification and a test resource specification. A design specification can be modelled as a

[7]: *design*, $D = (C, B, p_{max})$, where: $C = \{c_1, c_2, ..., c_n\}$ is a finite set of cores; each core, $c_i \in C$, is characterized by:

(x_i, y_i): placement denoted by x and y coordinates and each core c_i consists of a finite set of blocks $B_i = \{b_{i1}, b_{i2}, ..., b_{im}\}$ where $m > 0$.

Each block, $b_{ij} \in B_i$, is characterized by:

$minbw_{ij}$: minimal TAM bandwidth,

$maxbw_{ij}$: maximal TAM bandwidth.

p_{max}: maximal allowed power at any time;

Each testable unit is viewed as a block and several blocks form a core. An unwrapped core such as the UDL block in Figure 1 is modelled as a block at core A (bC in Figure 2). The wrapper at Core A will feed the test stimuli to the UDL block (bC) and core B will fed the test response from the UDL block to the TAM and therefore is coreB listed at *ict* for test C. The constraint list for test C include the blocks required for the testing bC (the UDL block) (Figure 2).

An advantage of having blocks within cores and a constraint list is that we can model testable blocks embedded within cores. Furthermore, it is possible to model test interference between different tests.

The core placement is given by (x,y) coordinates. We make use of a single point for each core since TAM routing is a complicated problem. For each block *minbw* and *maxbw* are given. For instance, the four scan chains at core A (Figure 1) can form one single *wrapper chain* and the minimal bandwidth is one. It is, however, also possible to form four wrapper chains making the maximal bandwidth equal to four. Five wrapper chains would not reduce the test time and therefore the designed will specify four as the maximum in this case.

A *test resource specification* [7], $TRS_k = (T, R_{source}, R_{sink}, source, sink)$ where each block, b_{ij}, at core, c_i, is attached with a finite set of tests, $T_{ij} = \{t_{ij1}, t_{ij2}, ..., t_{ijo_{ij}}\}$ and each test, $t_{ijo_{ij}} \in T_{ij}$, is characterized by:

$\tau_{ijo_{ij}}$: test time,

$p_{ijo_{ij}}$: test power,

$mem_{ijo_{ij}}$: required memory for test pattern storage.

$cl_{ijo_{ij}}$: constraint list with blocks required for the test.

$R_{source} = \{r_1, r_2, ..., r_p\}$ is a finite set of test sources where each test source, $r_i \in R_{source}$, is characterized by:

(x_i, y_i): placement denoted by x and y coordinates,

vbw_i : vector bandwidth,

$vmem_i$: size of vector memory.

$R_{sink} = \{s_1, s_2, ..., s_q\}$ is a finite set of test sinks; where each test sink, $s_i \in R_{sink}$, is characterized by:

(x_i, y_i): placement denoted by x and y coordinates,

rbw_i: response bandwidth,

source: $T \rightarrow R_{source}$ defines the test sources for the tests;

sink: $T \rightarrow R_{sink}$ defines the test sinks for the tests;

Each block (testable unit) can be tested by a set of tests where each test set is given by its test time, power consumption, memory requirement for test vector storage and a list of blocks required during testing.

```
# Example design
[Global Constraints]
MaxPower = 25
[Cores] name x    y    {block1, block2,..., block n}
        coreA 10  10   {bA, bC}
        coreB 20  10   {bB}
[Generators] name x    y    maxbw   memory
             r1   0    10     3     100
             r2   20   15     2     100
[Evaluators] name x    y    maxbw
             s1   30   10   4
             s2   20   5    4
[Tests] name  pwr time tg   tre  minbw   maxbw mem ict
        testA 10  90   r1   s1     1       2    50  no
        testB1 15 30   r1   s1     1       4    50  no
        testB2 5  50   r2   s2     1       4    50  no
        testC  15 10   r1   s2     1       2    20  coreB
[Blocks] name idle pwr {test1, test2,..., test n}
         bA     0    {testA}
         bB     0    {testB}
         bC     0    {testC}
[Constraints] test {block1, block2,..., block n}
              testA  {bA}
              testB1 {bB}
              testB2 {bB}
              testC  {bC, bA, bB}
```

Figure 2. An input specification to the test tool [7] of the example system in Figure 1.

The placement for each test sources and test sink is given by (x,y) coordinates. For test sources the placement, its maximal allowed bandwidth and the size of the memory are given and for test sinks the placement and the maximal allowed bandwidth are given. The functions, *source* and *sink*, connects a test with its test source and its test sink.

The input specification to the test tool, for the example system shown in Figure 1, is shown in Figure 2; it is the design specification and the test resource specification. For each design, we can have several specifications. Our problem is to select the test resource specification that leads to the best test solution.

4 Test Resource Selection

In this section we describe our estimation technique and how it is used to develop a test solution. The total cost of a test solution is given by the test application time and the amount of routed TAM wires. It can be computed as:

$$cost = \alpha \times \tau_{total} + \beta \times TAM \qquad 1$$

where τ_{total} is the test time (end time of the test with highest test time), *TAM* is the routing length of all TAM wires, and, α and β are user-defined constants used to determine the importance of test time in relation to TAM cost.

4.1 Estimation of Test Application Time

Each test can be illustrated by an “area” defined by its test time multiplied by its power consumption (Figure 3). The test time can often be modified by the assignment of a higher number of TAM wires. For instance, the four scan-chains in Figure 4 (a) (Core A in Figure 1) can form a single

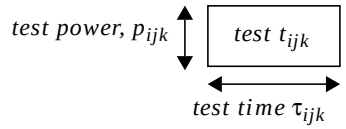

Figure 3. Test time and power consumption for a test.

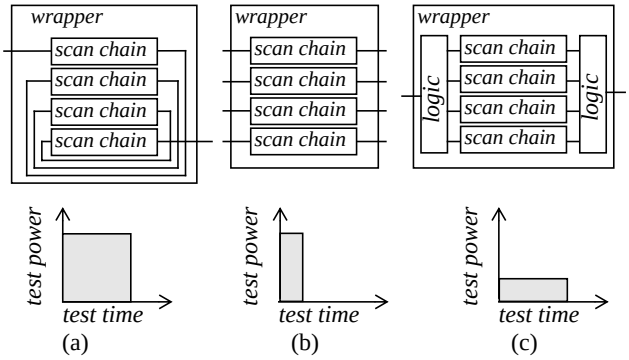

Figure 4. Core design alternatives.

wrapper chain connected to the single TAM wire. However, it is also possible, if several TAM wires are assigned that several wrapper chains are created as in Figure 4 (b). For a test, the test time depends on the number of wires [10]:

$$\tau'_{ijk} = \tau_{ijk} / (tam_{ijk}) \qquad 2$$

The test power consumption highly depends on the switching activity. Saxena *et al*. proposed a scan-chain gating scheme that reduces the test power consumption [13]. The advantage of the approach is that instead of having the same test power independent of the number of TAM wires as in Figure 4 (a) and Figure 4 (b) the test power consumption becomes dependent on the number of assigned TAM wires (see Figure 4 (b, c).) In Figure 4(c) logic is to be added, however, the usage and routing of TAM wires is reduced, which is of more importance [3]. The relation between test power and number of TAM wires [10]:

$$p'_{ijk} = p_{ijk} \times tam_{ijk} \qquad 3$$

where $k=o_{ij}$.

We now summarize the $p_{ijk} \times \tau_{ijk}$ for all testable units and by dividing by P_{max} we achieve the systems optimal test application time, which we use as an estimate, $\tau_{estimate}$ [10]:

$$\tau_{estimate} = \sum_{i=1,\, j=1,\, k=1} \frac{\tau_{ijk} \times p_{ijk}}{P_{max}} \qquad 4$$

where $k=o_{ij}$, τ_{ijk} is the test time and p_{ijk} is the test power consumption at block b_{ij} at core c_i, and P_{max} is the total power limit. The $\tau_{estimate}$ is computed ignoring test conflicts, however, we use it is an estimate of the systems test application time, *i.e.* we let $\tau_{total}=\tau_{estimate}$.

4.2 Estimation of TAM Cost

We estimate the length of a TAM wire using the city-block or Manhattan distance function $M(s,t)$:

$$M(s, t) = |x_s - x_t| + |y_s + y_t|$$

where s and t are two points at coordinates (x_s,y_s) and (x_t,y_t).

The length l_{ijk} of the TAM wires required for executing the test t_{ijk} is the summation of the length connecting the test source r to the wrapped core receiving test data c_i and connecting the wrapped core receiving the test response c_j to the test sink s. The distance is given by:

$$l_{ijk} = M(r, i) + M(j, s) \qquad 5$$

where c_i is the core receiving test vectors from test source r and c_j is the core sending the test responses to test source s. In the case of a test at a wrapped core, $c_i=c_j$.

The cost $cost_{ijk}$ for a test t_{ijk} ($k=o_{ij}$) can be formulated as:

$$cost_{ijk} = \alpha \times \tau_{ijk} / (tam_{ijk}) + \beta \times tam_{ijk} \times l_{ijk} \qquad 6$$

where α and β are the user-defined constants determining the relative importance of test time and TAM, τ_{ijk} is the test time, tam_{ijk} is the number of TAM wires assigned to the test and l_{ijk} is the length of the required TAM wires connecting the required test source with the testable unit and the test sink. For estimating the tam_{ijk} we derivate the cost function in respect to tam_{ijk} and get:

$$cost'_{ijk} = -\alpha \times \tau_{ijk} / (tam_{ijk})^2 + \beta \times l_{ijk} \qquad 7$$

and by letting $cost'_{ijk}=0$ we compute:

$$tam_{ijk} = \sqrt{\alpha \times \tau_{ijk} / \beta \times l_{ijk}} \qquad 8$$

We derivate a second time and get:

$$cost''_{ijk} = 2 \times \alpha \times \tau_{ijk} / (tam_{ijk})^3 + \beta \times l_{ijk} \qquad 9$$

and by letting $cost''_{ijk}=0$, we find that for $tam_{ijk}>0$ $cost''_{ijk}$ is always positive, *i.e.* a minimum.

The total TAM cost *TAM* is estimated as:

$$TAM = max\left\{ tam_{ijk} \times l_{ijk}, \sum_{i=1,\, j=1,\, k=1}^{|T|} \frac{tam_{ijk} \times l_{ijk}}{|T|} \right\} \qquad 10$$

where T is the set of tests, tam_{ijk} is given by Equation 8 and rounded off upwards and l_{ijk} is given by Equation 5.

In Equation 10 we divide with the number of tests ($|T|$) because the tam_{ijk} and l_{ijk} are computed as if dedicated TAM wires are required for each test. That is, there is not more than one test for any TAM wire; TAM wires are not shared. However, we assume that TAM wires can be shared. We use the *max* for cases when a test require long wires.

4.3 Example

The specification in Figure 2 is used to illustrate the cost estimation ($\alpha=\beta=1$). The estimated costs are in Table 1. The estimated test time is computed as the summation of the product *time×power* for each test divided by P_{max} (Eq. 4). For the example, the test time is computed to 70 . The wire length (Eq. 5) and the bandwidth (Eq. 8) are computed for each test and the TAM cost, which gives an estimate on the TAM cost (Eq. 10).

4.4 Resource Utilization

The technique above is useful for the cost estimation of each of the test specifications. Based on the ranking of the test specifications the SOC integrator selects the test

	testA	testB1	testB2	testC	Total
Time× power $\tau_{ijk} \times p_{ijk}$	90×10=900	30×15=450	50×5=250	10×15=150	70 (Eq.4)
Wire length l_{ijk} (Eq.5)	10+20=30	20+10=30	5+5=10	10+10=20	-
Bandwidth tam_{ijk}(Eq.8)	2	1	2	1	-
TAM $l_{ijkk} \times tam_{ijkk}$	30×2=60	30×1=30	10×2=20	20×1=20	60 (Eq.10)

Table 1. Cost estimation of the system in Figure 1.

specification with the lowest cost. However, the estimation technique does not consider test conflicts, and therfore a technique is needed for it. A technique is also needed to pinpoint bottlenecks, since it helps the SOC test integrator to determine which resource to modify in the iterative process when creating the test requirement.

A machine-oriented Gantt chart can be used to show the allocation of jobs on machines [2]. We will use such a chart where the resources are the machines and the tests are the jobs. The Gantt chart is in Figure 5 for the example in Figure 1 with the test specification in Figure 2 and each test is assigned to the test resources it requires. For instance, test B2 needs TG: r2 and TRE: s2.

An inspection of Figure 5 shows that TG:r2 and TRE:s2 are not limiting the solution. On the other hand, all resources that are used more than τ_{opt} are limiting the solution. The test source TG:r1 is the most critical one followed by the test sink TRE:s1. These two test resources are the obvious candidates for modification.

The test resource optimization algorithm (Figure 6) starts with estimation and ranking of the design possibilities (line 1) and the loop at line 2 terminates when an efficient test resource partitioning is found. At line 3, the best *available* of the initially ranked solutions is selected and a second loop starts at line 4. At line 5 a Gantt chart is created for the test resource analysis. The limiting resource is identified and at line 6 modifications are computed such that for each test at the limiting resource one modification is allowed. A modification means that the TAM is increased by one for each test, which reduces the test time of each test. The best new solution is selected and kept if better than the old solution. If no new solution can be found, current TRS is marked as used and the algorithm restarts with a new unmarked TRS at line 3.

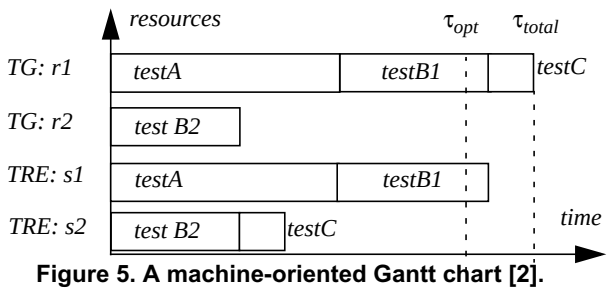

Figure 5. A machine-oriented Gantt chart [2].

```
1. evaluate and rank the TRS' (test resource specification)
2. until an efficient test resource partitioning is achieved begin
3.    select the best available TRS as initial solution
4.    until cost of best solution is acceptable begin
5.       create a Gantt chart of the TRS,
6.       for all tests at the limiting resource begin
7.             compute modification cost
8.       end
9.       if a better solution exists begin
10.         select the solution with lowest cost as new best solution
11.      end else begin
12.         mark the TRS as used
13.         restart with new TRS
14.      end
15.   end
16. end
```

Figure 6. Test Resource Optimization Algorithm

5 Experimental Results

We have made a comparison between the cost estimation technique proposed in the paper and our previously developed test scheduling and TAM design tool in [7]. We have created a set of test specifications and for each such specification we have estimated the cost and evaluated the cost using the tool in [7]. Note that even though we only created a few test specifications the number of possibilities is high and will increase rapidly for larger designs. We have made use of the design example with three testable units (Figure 1) named core A, B and C where C is the UDL block. For the cost estimation we have assumed that α=1 and β=1.

The design data, core placement and global power constraint of the system, is given in Figure 2. The test resources are specified in Table 2 and the tests are in Table 3. We have created 8 test specifications for the system (Table 4) and for each specification we have collected the data from the estimation and the test scheduling and TAM design tool (Table 5). We sorted the results based on the total cost from the tool. For the test specifications with the lowest total costs from the tool, our estimation technique works best, which is important since it is at these specifications the final test solution is likely to be found.

6 Conclusions

In this paper we have proposed a test resource partitioning and optimization technique that produces a test resource specification, the input to a test scheduling and TAM (test access mechanism) design tool. We have shown the complexity of the problem and proposed a technique at a low computational cost. We have also proposed a technique for the iterative improvement of a test specification by using Gantt charts. The advantage of our technique is that it pinpoints the bottlenecks in a test solution, which are the candidates for modification. We have also made a comparison with a test scheduling and TAM design tool.

Name	x	y	bw	Memory
TG.A1	0	10	10	100
TG.A2	0	10	5	50
TG.B1	10	10	10	0
TG.B2	10	10	10	25

Name	x	y	bw
TRE.A1	30	10	10
TRE.A2	30	10	5
TRE.B1	10	10	10
TRE.B2	20	10	10

Table 2. Test resources (A-ATE, B-BIST,bw-bandwidth).

Test	TG	TRE	Test time	Test power	Bandwidth [min,max]	M.	C.
TestA1	TG.A1	TRE.A1	50	15	[1,5]	50	-
TestA2	TG.A2	TRE.A2	30	10	[1,5]	25	-
TestA3	TG.B1	TRE.B1	40	10	[1,10]	0	-
TestA4	TG.B1	TRE.B2	50	15	[1,10]	10	-
TestB1	TG.A1	TRE.A1	60	10	[1,5]	20	-
TestB2	TG.A2	TRE.B1	60	10	[1,4]	5	-
TestB3	TG.B2	TRE.A1	30	15	[1,5]	0	-
TestC1	TG.B2	TRE.B2	5	5	[1,5]	10	A,B

Table 3. Test sets (M-memory, C-Constraint).

Test specification	Core A	Core B	Core C (UDL)
1	TestA1	TestB1	TestC1
2	TestA2,TestA3	TestB1	TestC1
3	TestA2, TestA4	TestB1	TestC1
4	TestA3, TestA4	TestB1	TestC1
5	TestA1	TestB2, TestB3	TestC1
6	TestA2,TestA3	TestB2, TestB3	TestC1
7	TestA2, TestA4	TestB2, TestB3	TestC1
8	TestA3, TestA4	TestB2, TestB3	TestC1

Table 4. Test specifications.

References

[1] K. Arabi, "Logic BIST and Scan Techniques for Multiple Identical Blocks", *Proceedings of VLSI Test Symposium (VTS)*, pp. 60-65, April 2002.

[2] P. Brucker, "Scheduling Algorithms", *Springer-Verlag*, ISBN 3-540-64105-X, 1998.

[3] A. L. Crouch, "*Design For Test*", Prentice Hall PTR, 1999.

[4] G. Hetherington *et al.*, "Logic BIST for Large Industrial Designs: Real Issues and Case Studies", *Proceedings of International Test Conference (ITC),* pp. 358-367, Sep. 1999.

[5] V. Iyengar and K. Chakrabarty, "Precedence-Based, Preemptive, and Power-Constrained Test Scheduling for System-on-a-Chip", *Proceedings of IEEE VLSI Test Symposium (VTS),* pp. 368-374, April 2001.

[6] G. Jervan, Z. Peng, R. Ubar, and H. Kruus, "A Hybrid BIST Architecture and its Optimization for SoC Testing", *Proceedings of International Symposium on Quality Electronic Design (ISQED'02)*, pp. 273-279, March 2002.

[7] E. Larsson, K. Arvidsson, H. Fujiwara, and Z. Peng, "Integrated Test Scheduling, Test Parallelization and TAM Design", *Proceedings of Asian Test Symposium (ATS)*, pp. 397-404, November 2002.

[8] E. Larsson and Z. Peng, "An Integrated Framework for the Design and Optimization of SOC Test Solutions", *Journal of Electronic Testing: Theory and Applications, (JETTA)*, vol. 18, pp 385-400, August 2002.

[9] E. Larsson, A. Larsson, and Z. Peng, "Linkoping University SOC Test Site", http://www.ida.liu.se/labs/eslab/soctest/

[10] E. Larsson and Z. Peng, "Test Scheduling and Scan-Chain Division Under Power Constraint", *Proceedings of Asian Test Symposium (ATS)*, pp. 259-264, November 2001.

[11] E. J. Marinissen *et al.*, "On IEEE P1500's Standard for Embedded Core Test", *Journal of Electronic Testing: Theory & Applications, (JETTA)*, vol. 18, pp 365-383, August 2002.

[12] S. Mourad and Y. Zorian, "Principles of Testing Electronic Systems", *John Wiley & Sons,* ISBN 0-471-31931-7, 2000.

[13] J. Saxena, K. M. Butler, and L. Whetsel, "An Analysis of Power Reduction Techniques in Scan Testing", *Proc. of International Test Conference (ITC)*, pp. 670-677, Oct. 2001.

[14] M. Sugihara, H. Date, and H. Yasuura, "Analysis and Minimization of Test Time in a Combined BIST and External Test Approach", *Proceedings of Design and Test in Europe (DATE),* pp. 134-140, March 2000.

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

Test specification (Table 4)	Test time (τ)			TAM cost (c)			Total cost ($t=\alpha\times\tau+\beta\times c$)		
	Estimation, τ_e	Tool [7,9] τ_t	Difference (%) $\lvert 100\times(\tau_t-\tau_e)/\tau_t\rvert$	Estimation c_e	Tool [7,9] c_t	Difference (%) $\lvert 100\times(c_t-c_e)/c_t\rvert$	Estimation t_e	Tool [7,9] t_t	Difference (%) $\lvert 100\times(t_t-t_e)/t_t\rvert$
1	55	65	15.4	60	60	0	115	125	8.0
2	59	75	21.3	60	60	0	119	135	11.9
4	71	95	25.3	60	40	50.0	131	135	3.0
3	73	85	14.1	60	70	14.3	133	155	14.2
5	73	95	23.2	60	80	25.0	133	175	24.0
6	77	95	18.9	60	80	25.0	137	175	21.7
8	89	95	6.3	60	90	33.3	149	185	19.5
7	91	115	20.9	60	120	50.0	151	235	35.7
Average:			18.2			24.7			17.2

Table 5. Experimental results sorted based on final total cost.