Test Infrastructure Design and Test Scheduling Optimization

Erik Larsson and Zebo Peng
Dep. of Computer and Information Science Linköping University, SE-581 83 Linköping, Sweden
Phone: +46 13 284046, Fax: +46 13 282666, Email: {erila, zpe}@ida.liu.se

Abstract[1]

We propose a technique for test scheduling and design of test bus infrastructure where test application time as well as test bus length and width are minimized while constraints on power consumption and test resources are considered. Our approach is suitable for repeated use in the design space exploration process due to its low computational cost. For the final design, we use simulated annealing to optimize the solution. The proposed technique is implemented and experimental results show the efficiency of our approach.

1. Introduction

Test bus infrastructure design and test scheduling are becoming more crucial due to the increasing complexity of digital designs. A test schedule is used to determine the order of the tests. Zorian proposes a dedicated test controller for a BIST system [2]. A more general test scheduling algorithm is proposed by Chou *et al.* where test application time is minimized while power consumption is kept under control [1].

We propose a technique to combine test scheduling and test bus infrastructure design. Our technique minimizes test application time as well as test bus length and width while considering constraints on power consumption, BIST resources and cores. Our approach considers both internal tests of the cores as well as interconnection tests between the cores. Furthermore, each core may be tested by several tests using different test facilities where the BIST resources may be shared among several tests at different cores. The low computational cost of our technique allows it to be used repetitively in the design space exploration. For the final design, we use simulated annealing which is a more extensive optimization technique.

2. Preliminaries

We assume a Design for Test (DFT) flow where an architecture is created from a system-level specification. It is made testable by the introduction of DFT techniques. After that, the tests are scheduled and a test bus infrastructure is added to transport test data in the system, which is the focus of this paper.

If the design, at any step in the DFT flow, shows unaccepted results, some design steps have to be repeated. Therefore it is important that the algorithms used in the design space exploration process has a low computational cost. Before the final design is generated, on the other hand , a more extensive optimization can be allowed to take longer time, *i.e* have higher computational cost.

We assume that the system uses a test infrastructure strategy as proposed by Marinissen *et al.* [4].

A system can be tested by external testers or with BIST techniques or a combination of both. If an external tester for a scan-based design is used the test vectors is transported from a test access port (TAP) to the scan chains and the test response is transported from the scan-chains to a TAP.

On the other hand if a core uses BIST, the test vectors have to be transported from its test generator to the core and from the core to its signature analyzer. The test generator and signature analyzer can be placed anywhere in the system which is desirable if several cores share a BIST infrastructure.

3. Test Scheduling and Bus Design Algorithms

The algorithm for combined test scheduling and test bus design is based on the extension of our previously proposed test scheduling algorithm [3] and is illustrated in Figure 1.

In general the algorithm works as follows. All tests in the system are sorted and placed in a list according to power consumption. A loop at step 3 is repeated until all tests are scheduled. In each iteration, the test consuming most power is checked and if it can be scheduled according to the power, BIST and core constraints it is removed from the list. A test bus is then selected or designed and the test is scheduled.

At step 7 in Figure 1 the power, BIST and core constraints are checked if a test is allowed to be scheduled, the checking procedure is found in Figure 2. If all constraints are satisfied, the test is scheduled.

The test bus design is performed at step 8 in Figure 1 and the test bus design algorithm is presented in Figure 3. Among the not scheduled buses at a certain time, where all of the test buses connecting all resources for the selected test, the test

```
 1: Sort the tests according to power consumption;
 2: Place the tests in a list with the
    test consuming most power first;
 3: while not empty list
 4:    cur = first in list;
 5:    while not at end of list do begin
 6:       pick cur from list;
 7:       if constraints = true then
 8:          bus = test_bus(cur,time,schedule)
 9:          schedule(cur, time, bus)
10:          remove_from_list(cur);
11:       cur = next_unit;
12:   time=time+min(scheduled units);
```

Figure 1: Test scheduling algorithm.

1. This work has partially been supported by NUTEK.

```
constraints ( test, time, schedule)
return = false;
 if (avail_power+test power < Max power )
   if ( Signature_Analyzer( test ) = free )
      if ( Test_Generator( test ) = free )
         if ( Core_constraint( test ) = free )
            return_value = true;
return return_value;
```

Figure 2: Constraint checking.

```
test_bus( test, time, schedule){
if exists_free_bus( time )
   if exists_free_used_bus( time )
      select the shortest free;
   else select the shortest;
else
   select bus requiring shortest extension
   until test connected
      extend and update shortest part;
```

Figure 3: Algorithm for Test bus design.

bus that has the shortest length and has been used before in the schedule is selected. If none of the test buses has been used before, the shortest is selected.

If no test bus is free for scheduling and connecting all resources in the test, a test bus is extended or a new is added. Among all buses in the system, the one that requires the shortest extension is selected.

4. Test Bus Design and Test Scheduling Tool

Our tool accepts design specification as in Figure 4 where the maximal allowed power consumption is specified and for each core several tests can be defined. Each test is placed at a core and an idle power consumption, a test power consumption and a test application time are specified. Furthermore, for each test several constraints can be defined.

The core constraints are introduced for two reasons. At interconnection tests where the interconnection between two cores is tested both core wrappers must be in inter connection mode. The wrapper isolates the core from the rest of the system when a test is applied. However, it does not isolate tests internally at a core. When several tests are defined at a core, they could interfere with each other and we can not schedule these tests concurrently. In the specification to our system this is indicated by placing the core name both as the core name and as the core constraint

5. Experimental Results

We compare our approach with a traditional technique which performs test scheduling and test bus design in two separate steps. We have also an implementation based on Simulated annealing optimization [5].

The cost function $C(x)$ for a solution x, where $T(x)$ is the test application time, $L(x)$ is the test bus length and, α is a factor defined by the designer:

$$C(x) = T(x) + \alpha \times L(x)$$

The experimental results are in Table 1. The cost of the solution generated by the traditional approach is 980 for ASICZ [2] while the combined approach uses 870, an improvement of 11.2%. The simulated annealing produces a

```
// Max power: 250
//core plac. test  idle test test TG  SA   core
//name x  y  name  pwr  pwr  time
core1  10 10 test1 10   100  125  TC  TC   NONE
core1  10 10 test2 20   200   75  TG1 TC   core2
core2  10 20 test3 50   150   50  TC  SA1  NONE
core2  10 20 test4 50   100  100  TG1 SA1  NONE
core3  10 30 NONE
TC     20 10 NONE
TG1    20 20 NONE
SA1    20 30 NONE
```

Figure 4: A design example accepted by our design tool.

solution at a cost of 662 which is 32.4% better than the traditional solution. The traditional and combined solutions are found within a second while the solution from simulated annealing is found after 3543 seconds or 88000 iterations.

We generated designs with 4, 9, 16 and 25 cores with test data randomly generated. For each size, 5 designs were created and the average result is presented.

The results in Table 1 shows that the combined algorithm finds a solution at a very low computational cost, less than one second for all designs. The simulated annealing finds a better solution in all cases and the improvement is larger as design complexity increases.

Design	Traditional		Combined		Cost Impr. (%)	Simulated annealing			Cost Impr. (%)
	Cost	CPU(s)	Cost	CPU(s)		Av. cost	Cpu(s)	Itera-tions	
ASICZ	980	<1	870	<1	11.2	662	3543	88000	32.4
d4	260	<1	242	<1	6.9	211	30633	88000	18.8
d9	637	<1	607	<1	4.7	435	968	88000	31.2
d16	1235	<1	1212	<1	1.9	760	3822	88000	38.5
d25	2477	<1	2383	<1	3.8	1436	16630	88000	42.0

Table 1: Experimental results comparing separate solution, combined solution and simulated annealing.

6. Conclusions

We have proposed an approach for combined test scheduling and test bus design where the test application time and the test bus infrastructure are minimized while constraints on the power consumption, BIST resources and core constraints are considered. Due to its low computational cost our technique is suitable to be incorporated in the design space exploration process. A more extensive optimization based on simulated annealing is used for the final design.

References

[1] R. Chou, K. Saluja, V. Agrawal, Scheduling Tests for VLSI Systems Under Power Constraints, IEEE Trans. on VLSI Systems, Vol. 5, No. 2, page 175-185, June 1997.

[2] Y. Zorian, A distributed BIST control scheme for complex VLSI devices, VLSI Test Symposium, April 1993.

[3] E. Larsson, Z. Peng, An Estimation-based Technique for Test Scheduling, Proc. of Electronic Circuits and Systems, pp 25-28, Bratislava, September, 1999.

[4] E. J. Marinissen, et al, A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores, Proceedings of ITC October 1998.

[5] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, Science 220(4598), pp 671-680, 1983.