An Estimation-based Technique for Test Scheduling

Erik Larsson and Zebo Peng
Dep. of Computer and Information Science Linköping University, Sweden
Email: {erila, zpe}@ida.liu.se

Abstract[1]

A technique to schedule tests for complex digital systems is proposed where the test application time is minimized while the power dissipation is kept under control. The technique is based on estimations of the test application time and the power consumption. By experiments we show the efficiency of our approach.

1. Introduction

Development of microelectronic technology has lead to the implementation of system on chip (SOC), where a complete system, consisting of several ASICs, microprocessors, memories and intellectual properties (IP) blocks, is placed on a single chip. Such a system is usually made testable by the introduction of some design-for-testability (DFT) mechanism. Several DFT techniques such as test point insertion, scan and different types of built-in self-test (BIST) have been developed[1]. For a complex SOC design it is not unusual to combine several test techniques since they all have their respective advantages and disadvantages. Furthermore, when IP blocks are used, they may already contain a test methodology which is different from the rest of the design and it has to be incooperated in the overall test strategy of the whole system.

There are many similarities in testing PCB (printed circuit boards) and SOC. The major difference is however two-fold. In PCB, testing of each individual component can often be carried out before mounting and the components can be accessed for test via probing. Neither of these is possible when testing SOC. This means that testing the completed system becomes even more crucial and difficult. In order to keep test application time at a minimum, it is desirable to apply as many tests as possible concurrently. However, the power consumption must be kept under control. Otherwise, the chip could be damaged. Furthermore, due to resource conflicts, it is usually not possible to apply all tests concurrently.

Much research has focused on the scheduling problem in digital system design [6]. The main question has been to find a schedule that fulfills a set of constraints and minimizes a given cost function. In high-level synthesis the cost function has traditionally been an area performance function. Today, new approaches have been proposed where issues such as testability and power consumption are considered. In scheduling for low power, the scheduling is performed to find a design which minimizes the power consumption. These techniques focus on the normal mode and not at the test mode. The power consumption at the test mode can be much higher than during normal operation due to the high switching activity [2]. Therefore, it is very important to consider power consumption in the test mode. A major problem in synthesis for low power is that the switching activity depends on the input data. Several techniques have been proposed to estimate the input data.

A schedule has to be developed to determine the order of the tests. A dedicated test controller for a BIST based system is proposed by Zorian [5]. A general test scheduling algorithm is proposed by Chou *et al.* [4] and another test scheduling approach is proposed by Stroele *et al.* [3].

In this paper we assume a design with added DFT features. We propose a technique to estimate the test application time and the power consumption. Our approach is based on estimating the needed test vectors for the design. We propose a test scheduling technique which is based on these estimations. Our scheduling technique minimizes the test application time while the power dissipation is kept under control.

The rest of the paper is organized as follows. First we present the technique to determine which test will be used to test a resource in Section 2. In Section 3 the estimation of test application time is presented and the technique to estimate the test power consumption is given in Section 4. Our scheduling technique is presented in Section 5. Experimental results are in Section 6 and finally we conclude the paper in Section 7.

2. System Model and Test Strategy

A digital system can be seen as a set of interconnected blocks. To test it, a set of tests is applied and to minimize the test application time all tests should be applied as concurrently as possible. However, it might not be possible to apply

---

1. This work has partially been supported by the Swedish National Board for Industrial and Technical Development (NUTEK).

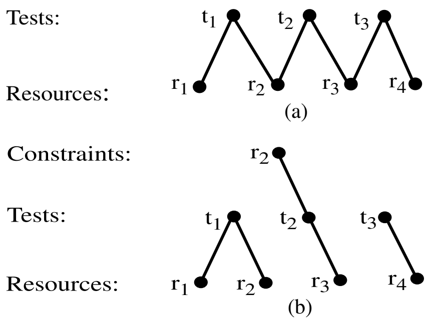

Figure 1: Tests, resources, and constraints.

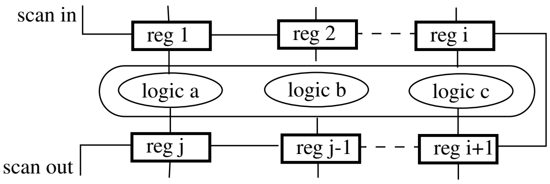

Figure 2: A Scan example.

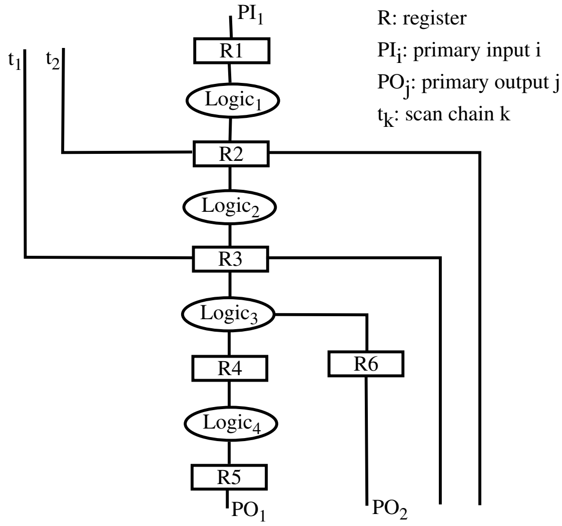

Figure 3: An example to illustrate shortest controllable and observable paths.

the test concurrently due to high power dissipation which might damage the circuit under test. Furthermore, there might exist different types of conflicts among tests and resources that restrict the ability for concurrent testing.

A resource graph can be used to describe the relationship between the tests and the IP blocks [4]. An example of resource graph is in Figure 1(a) where an edge from test t_1 to resource r_1 indicates that t_1 tests r_1. In Figure 1(a) test t_1 and test t_2 can not be executed at the same time since both tests resource r_2. In practice, we note that an edge, connecting t_i and r_j, in a resource graph could actually mean one or both of the following situations:

- A test t_i tests a resource r_j.
- When a test t_i is applied, resource r_j is required for test isolation and/or test access.

We introduce the *constrained resource graph* where a *constraint* level is added in order to distinguish between the two cases. If a resource is needed for other reasons but testing it is placed on the constraint level. For instance if resource r_2 in Figure 1(a) is needed in order to apply test t_2, we place it on the constraint level, see Figure 1(b).

Our second observation is that a resource has to be tested only once. We introduce a technique to determine which test should be used to test a resource. For instance resource r_3 in Figure 1(a) can be tested by either t_2 or t_3. In Figure 3 $logic_3$ can be tested using the primary input PI_1, scan chain t_1, or scan chain t_2. If PI_1 is used for the test, the test vectors have to pass all the logic from PI_1 to $logic_3$. Furthermore, if PI_1 is used for the test of $logic_3$ neither scan chain t_1 or scan chain t_2 can be used concurrently. On the other hand, if scan chain t_1 is used to test $logic_3$ the test vectors have to pass through less logic and PI_1 and t_2 can be used to test other units at the same time. We use the *Shortest Controllable Path* and *Shortest Observable Path* to determine which test t_i should be used to test a resource r_j [8]. Let $G(V, E)$ be a directed graph where a vertex, $v \in V$, corresponds to a functional unit (operation). A *start vertex* is a vertex which gets its value directly from a primary input or a test access port. An *end vertex* is a vertex connected to a primary output or a test access port. A path, P_i, is a sequence of edges $\{(v_0, v_1), (v_1, v_2), ..., (v_{n-1}, v_n)\}$ between v_0 and v_n. The *Shortest Controllable Path*, $SCP(op)$, for an operation *op* is the shortest path from a start vertex to *op*, and the *Shortest Observable Path*, $SOP(op)$ is the shortest path from an operation *op* to an end vertex.

For instance, in Figure 3, $SCP(logic_3) = \{(t_1, R_3), (R_3, Logic_3)\}$ and $SOP(logic_3) = \{(Logic_3, R_6), (R_6, PO_2)\}$. The first element on the *SCP* is where the test vectors are applied. The constraints in the constrained resource graph are the vertices on the *SCP* and *SOP*. According to our approach, $logic_3$ is tested via t_1 and R_3, R_6 and PO_2 are in the set of constraints.

3. Estimation of Test Application Time for the Scan Technique

An operator is tested by applying a set of test vectors on the inputs and observe the response on the outputs of the operator. The test application time depends mainly on the number of test vectors and the clock frequency. The total test application time $T(t_i)$ for a test t_i is given by:

$$T(t_i) = T_c(t_i) + T_a(t_i) + T_o(t_i) \qquad (1)$$

where T_c is the time to set up the test (*control*), T_a is the time to *apply* the test and T_o is the time needed to *observe* the test response. For the scan technique we use the following definition:

f_t: *clock frequency at test mode,* f_n: *clock frequency at normal mode,* n: *number of clock cycles in normal mode.*

$t_v(op)$: *number of test vectors needed to test operation op.*

$l_{in}(t_i, op)$: *length of scan chain* t_i (*number of flip-flops*) *from the input to operation op.*
$l_{out}(t_i, op)$: *length of scan chain* t_i (*number of flip-flops*) *from operation op to the output.*
$Op(t_i)$: *the set of operators tested by test* t_i.
$max(t_i, t_v(op))$: *the maximal number of test vectors needed of any operator op tested by* t_i *where* $op \in Op(t_i)$.

The test setup (control) time T_c , the application time T_a and the observation time T_o for a test t_i is estimated to:

$$T_c(t_i) = \frac{l_{in}(t_i, op)}{f_t} \times max(t_i, tv(op)) \qquad T_a(t_i) = \frac{n}{f_n} \times max(t_i, tv(op)) \qquad T_o(t_i) = \frac{l_{out}(t_i, op)}{f_t} \times max(t_i, tv(op))$$

where $op \in Op(t_i)$.

An example of a scan-based system is given in Figure 2. Assume that j=8, the bit width of all registers to be 16 bits, n=1 and f_n=f_t=10 MHz, and *logic a* requires 100 test vectors, *logic b* 50 and *logic c* 10. Using the above formulae the estimated test application time is: $100*(32+32+1)/10^6$=0.65 ms.

In the above formulae we assume that we know in which position of the scan chain we have the operator to be tested by the vectors. If that is not known we have to scan in the whole vector for the scan chain before a test can be performed[1]. In this case, instead of using $l_{in}(t_i)$ and $l_{out}(t_i)$ we use: $l(t_i)$: *length of scan chain* t_i (*number of flip-flops*).

The total estimated test application time becomes:

$$T(t_i) = \left(\frac{n}{f_n} + \frac{l(t_i)}{f_t}\right) \times max(tv(op))$$

where $op \in Op(t_i)$.

Taking the example in Figure 2 with the assumptions above, we estimate test application time to: 1.29 ms.

4. Estimation of Test Power Consumption

The power dissipation in a CMOS circuit consists of a static and a dynamic part. The static power dissipation due to leakage current or other current drawn continuously from the power supply, and the dynamic power dissipation is due to switching transient current and charging and discharging of load capacitances [7].

The static power dissipation and the dissipation due to switching transient current are negligible compared to the dissipation due to load and unload of capacitances. The power dissipation due to charging and discharging capacitances is given by [7]:

$$P_{dyn} = \frac{1}{2} \times V^2 \times C \times f \times a \qquad (2)$$

where V is the voltage, C is the capacitance, f is the frequency and a is the switching activity.

All parts but the switching activity in the formula 2 are given or can be extracted from a design library. The switching activity depends on the input data and there exists mainly two approaches to estimate it, based on simulation or probability. During testing the input to the design is known. It is the test vectors. We can for each component in a design use the test vectors generated by an ATPG tool as the input to a power simulation tool and then build a library with test power consumption for each unit in the design.

5. Test Scheduling Algorithm

In this section we propose a test scheduling algorithm which minimize the test application time while keeping power dissipation under control. It assumes that the test application time and the test power dissipation have been estimated for all units.

If two blocks are scheduled for test starting at the same time it is likely that one of the tests is completed before the other because the tests are of unequal length. This releases resources and a new test can be scheduled. However, if all tests in a test session are fully executed before tests from the next session is scheduled, the complexity of the test controller is minimized [4]. We present a test scheduling algorithm in Figure 4. The variable *time* in the algorithm refers to test session number.

At steps 1 and 2 in the algorithm we sort the tests according to estimated power consumption, see Figure 4. At step 4 we have a loop which terminates when all tests are scheduled and at step 7 we have an embedded loop where we try to schedule as many tests as possible at a specific test session. At step 8, we check if we can schedule the current test at this test session. We make sure that the power consumption is kept under the limit and we check the constrained resource graph in order to avoid conflicts between tests at the present test session.

In order to handle unequal test length the following step is modified:

```
15: time = time + min(scheduled units);
```

At step 15 we set time for next schedule to be the time when the shortest scheduled test is completed.

1. It is obvious that scan in and scan out can be performed simultaneously. However, here we are interested in the relative test application time for different blocks in the design. Therefore, we use an approach with low computation complexity.

```
Sort the tests according to (estimated)
power consumption;
Place the tests in a list where the test
consuming most power first;
time=0;
while not empty list
 pwr(time)=0;
 cur=first_test;
 while not at end of list do begin
    if pwr(time)+pwr(cur)<Max_pwr
    and no constraint(cur, scheduled)
    then
      schedule(cur, time)
      remove_from_list(cur);
      pwr(time)=pwr(time)+pwr(cur);
    end;
    cur=next_test;
  end;
  time=time+1;
end;
```

Figure 4: Test scheduling algorithm.

Block	Size	PB/active (mW)	PB/idle (mW)	Test Length
RL1	13400 gates	295	-	134
RL2	16000 gates	352	-	160
RF	64*17 bits	95	19	10
RAM1	768*9 bits	282	20	69
RAM2	768*8 bits	241	17	61
RAM3	768*5 bits	213	11	38
RAM4	768*3 bits	96	7	23
ROM1	1024*10 bits	279	23	102
ROM2	1024*10 bits	279	23	102

Table 1: Power dissipation and estimated test length for ASIC Z.

Test session	Zorian		Chou et al.		Our approach	
	Time	Blocks	Time	Blocks	Time	Blocks
1	69	RAM1, RAM4, RF	69	RAM1, RAM3, RAM4, RF	160	RL2, RL1, RAM2
2	160	RL1, RL2	160	RL1, RL2	102	RAM1 ,ROM1, ROM2
3	61	RAM2, RAM3	102	ROM1, ROM2, RAM2	38	RAM3, RAM4, RF
4	102	ROM1, ROM2	-	-	-	-
Tot. Time:	392		331		300	

Table 2: A comparision of different test scheduling approaches.

6. Experimental Results

We have used the ASIC Z [5] example to compare the scheduling algorithm presented in this paper with other approaches. The example consists of 9 blocks, see Table 1, where the estimation of the test length is made by Chou *et al.* based on the size of the blocks [4]. We use the same assumptions as Chou *et al.*, which is maximal power dissipation limited at 900 mW, all tests can be applied concurrently, the power consumption for idle blocks are excluded, and no tests can be started until all tests in the previous session is completed. In Table 2 we compare our test scheduling technique with Zorian's solution [5] and the approach proposed by Chou *et al*. [4]. Using our approach the total test length (application time) is 300 while the approach proposed by Zorian's has 392 and the Chou *et al.'s* approach has 331.

7. Conclusions

The high complexity in digital design has increased the need for design for testability techniques and the need to combine several DFT techniques. In order to minimize the test application time while keeping the test power consumption under control, efficient test scheduling is required. In this paper, we have discussed how to estimate the test application time and the test power consumption. We propose a test scheduling technique and experimental results show the efficiency of our approach.

References

[1] M. Abramovici, M. Breuer, A. Friedman, Digital Systems Testing and Testable Design, IEEE Press, ISBN 0-7803-1062-4, 1990.

[2] A. Hertwig, H-J Wunderlich, " Low Power Serial Built-In Self-Test ", *Proc. of ETW*, pp 49-53, Sitges, May, 1998.

[3] A. Stroele, H-J Wunderlich, " A Unified Method for Assembling Global Test Schedules ", *Proc. of the Third Asian Test Symposium*, pp 268-273, Nov. 1994.

[4] 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.

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

[6] D. Gajski, N. Dutt, A. Wu, S. Lin, High-Level Synthesis, Introduction to Chip and System Design, Kluwer Academic Pub., ISBN 0-7923-9194-2, 1992.

[7] C. Svensson, D. Liu, Low Power Circuit Techniques, in Low Power Design Methodologies, Kluwer Academic Publisher, ISBN 0-7923-9630-8, 1996.

[8] E. Larsson, Z. Peng, " Early Prediction of Testability by Analyzing Behavioral VHDL Specifications ", *Proc. of NORCHIP*, pp 259-266, Tallinn, November, 1997.