A COMBINED TREE GROWING TECHNIQUE FOR BLOCK-TEST SCHEDULING UNDER POWER CONSTRAINTS

Valentina Muresan, Mircea Vlăduțiu
"Politehnica" University of Timișoara, România
vmuresan@cs.utt.ro

Dublin City University, Ireland
muresanv@eng.dcu.ie

ABSTRACT

A tree growing technique is used here together with classical scheduling algorithms in order to improve the test concurrency having assigned power dissipation limits. First of all, the problem of unequal-length block-test scheduling under power dissipation constraints is modeled as a tree growing problem. Then a combination of list and force-directed scheduling algorithms is adapted to tackle it. The goal of this approach is to achieve rapidly a test scheduling solution with a near-optimal test application time. This is initially achieved with the list approach. Then the power dissipation distribution of this solution is balanced by using a force-directed global priority function. The force-directed priority function is a distribution-graph based global priority function. A constant additive model is employed for power dissipation analysis and estimation. Based on test scheduling examples, the efficiency of this approach is discussed as compared to the other approaches.

1. INTRODUCTION

It is well known that VLSI devices running in test mode consume more power than when running in normal mode [1]. Thus, one of the major considerations in test scheduling is the fact that the heat dissipated during test application is much higher than during normal mode. Test scheduling is strongly related to test concurrency. Test concurrency is a design property which impacts testability and power dissipation. To satisfy high fault coverage goals with reduced test application time under certain power dissipation constraints, the testing of all components on the system should be performed in parallel to the greatest extent possible.

This paper focuses on the high-level power-constrained block-test scheduling problem. As the device technologies such as VLSI and MCM become mature, the problem lacks of practical solutions. An efficient scheme for overlaying the block-tests, called extended tree growing technique, is employed here together with a combination of classical scheduling algorithms in order to search for power-constrained block-test schedule profiles in a polynomial time. The algorithm fully exploits perfect parallelism under power dissipation constraints. This is achieved by overlaying the block-test intervals of compatible subcircuits to test as many of them as possible concurrently so that the maximum accumulated power dissipation is balanced and does not go over the given limit. This overlaying process is modeled with the help of the extended tree growing technique. A combination of list and force-directed scheduling algorithms, proposed before, is used together with the extended tree growing technique in order to search for near-optimal test scheduling solutions. A constant additive model is employed for power dissipation analysis and estimation throughout the algorithm.

2. TEST SCHEDULING PROBLEM

The elements of the test model, for which test schedules are sought, are enumerated next. The components which are required to perform a test (test control logic, test buses, test pattern generators, signature analyzers, blocks under test (BUT), and any intervening logic) are known as test resources and they may be shared among BUTs. Each activity or the ensemble of activities requiring a clock period during the test mode and occurring in the same clock period, can be considered as a test step. A block-test is the sequence of test steps that correspond to a specific part of hardware (block). The testing of a VLSI system can be viewed as the execution of a collection of block-tests. The steps in a step sequence belonging to the same block-test can be pipelined and steps from different block-tests can be executed concurrently, obviously if there are no resource conflicts between the steps.

Block-tests and test steps have their resource sets used to build up their test plans. Depending on the test design methodology selected, once a resource set is compiled for each test t, then it is possible to determine whether they could run in parallel without any resource conflict. A pair of tests that cannot be run concurrently is said to be incompatible. Each application of time compatible tests is called a test session, and the time required for a test session is named test length. Two major types of test parallelism approaches have been identified in the literature thus far: block-test scheduling, which deals with tests for blocks of logic, and test pipelining, which deals with test steps that need to be applied and resources to be utilized in a specific temporal order.

If \( p(t) \) is the instantaneous power dissipation during test \( t \), and \( p(t) \) is the instantaneous power dissipation during test \( t \), then the power dissipation of a test session consisting of just these two tests is approximately \( p(t) + p(t) \). Usually this instantaneous power is constrained to not exceed the power dissipation limit, \( P_{max} \), if they were meant to be executed in the same test session. In order to simplify the analysis, a constant additive model is employed for power estimation. That is, a constant power dissipation value \( P(t) \) is associated with each block-test \( t \). For high-level approaches the power dissipation \( P(t) \) of a test \( t \) could be estimated in three ways: average power dissipation, maximum power dissipation, and RMS power dissipation over all test steps in \( t \). The total power dissipation at a certain moment of the test schedule is computed by simply summing the power dissipation of the running block-tests. The power dissipation \( P(s) \) for a test session \( s \) can be defined as: \( P(s) = \sum_{t \subseteq s} P(t) \), while the power constraint in test scheduling is defined as: \( P(s) \leq P_{max} \). The constant power dissipation value \( P(t) \) considered here is the maximum power over all test vectors applied in test \( t \).
3. PROPOSED APPROACH

Power dissipation during test, especially at high level, was seldom under research so far. Some approaches tackled the power dissipation problem during test application at low-level: switching activity conscious ATPG, scan latch reordering, test vector reordering, and test vector inhibiting. The BIST scheduling approach given in [1] is one of the first to take into account the power dissipation during test scheduling at block level. It performs global optimization considering also other factors such as block type, adjacency of blocks (device floor plan), but the latter are hardly known at high-level. [2] makes for the first time a thorough theoretical analysis of the power-constrained test scheduling (PTS) problem at IC level. It proposes a compatible test clustering technique which is an NP-complete approach.

Instead, a greedy approach is proposed in this paper. It has a polynomial complexity, which is very important for the success of the system-level test scheduling problem. It is a sequence involving list and force-directed scheduling algorithms adapted to work with a growing tree like modeling of the power-constrained test scheduling process. The proposed algorithm deals with tests of blocks of logic, which do not have equal test length. Thus, it is an unequal-length block-test scheduling. It is meant to be part of a system-level block-test approach to be applied on a modular view of a test hierarchy. The modular elements of this hierarchy could be: subsystems, backplanes, boards, MCM's, IC's (dies), macro blocks and RTL transfer blocks. Every test node \( t \) is characterized by a few parameters, which it has previously been assigned with, after the test scheduling optimization has been applied on it. These features are: test application time \( T_i \), power dissipation \( P_i \), and test resource set \( R.E.S.S.E.T_i \). This approach assumes a bottom-up traversing of the hierarchical test model within a divide et impera optimization style.

![Figure 1: Merging Step Example](image)

3.1. TREE GROWING TECHNIQUE

The tree growing technique has been proposed before in [3, 4]. However, it will be explained again for paper clarity reasons. The block-test set is huge in complex VLSI circuit designs and ranges in test length. Thus, it is possible to schedule some short tests to begin (if they are resource and power dissipation compatible) when subcircuits with shorter testing time have finished testing, while other subcircuits with longer testing time have not. Therefore, the test scheduling discipline assumed here is the partitioned testing with run to completion defined in [5]. The tree growing technique given in [3] is very productive from this point of view. That is because it is used to exploit the potential of test parallelism by merging and constructing the concurrent testable sets (CTS).

This was achieved by means of a binary tree structure (not necessarily complete), called compatibility tree, which was based on the compatibility relations among the tests.

A big drawback in [3] is that the compatibility tree is a binary one. This limits the number of children test nodes that could be overlapped to the parent test node to only two. In reality the number of children test nodes can be much bigger, as in the examples depicted in figures 1 and 2. Therefore an expanded compatibility tree (ECT), given by means of a generalized tree, is proposed instead to overcome this problem. Figure 2 gives the test scheduling chart and the ECT for the first test scheduling example presented in [4]. Figure 3 depicts the power-test scheduling chart for the scheduling solution generated by the proposed approach when there are no power constraints. In figure 2, the sequence of test nodes contained in the same tree path represents an expansion of the CTS. Given a partial scheduling chart of a CTS, a test \( t \) can be merged in this CTS if and only if there is at least one tree path \( P \) in the corresponding compatibility tree of CTS, such that every test contained in the nodes of \( P \) is compatible to \( t \). The compatibility relation here has three components. Firstly, tests have to be compatible from a conflicting resources point of view. Secondly, the test length of the nodes in a tree path have to be monotonously growing from leaf to root. Thirdly, the power dissipation accumulated on the above tree path should be less than or equal to \( P_{max} \).

![Figure 2: Tree Growing Example](image)

A merging step example is given in figure 1. Partial test schedule charts are given at the top, while partially grown compatibility trees are given at the bottom. Suppose tests \( t_2, t_3, t_4 \) are compatible to \( t_1 \), while they are not compatible to each other. Suppose \( T_1, T_2, T_3 \) and \( T_4 \) are, respectively, the test lengths of tests \( t_1, t_2, t_3 \) and \( t_4 \), and say \( T_2 + T_3 < T_1 \). Suppose now, a new test \( t_4 \) has to be scheduled in parallel to the partial test schedule depicted in figure 1(a). As can be seen, there is a gap \( GAP_3 \) given by the test length difference: \( GAP_3 = T_1 - (T_2 + T_3) \). Thus a merging step can be achieved, if \( T_4 \leq GAP_3 \), by inserting \( t_4 \) in the partial test schedule and its associated ECT in figure 1(b).

The process of constructing CTS's can be implemented by expanding (growing) the ECT from the roots to their leaf nodes. The root nodes are considered test sessions, while the expanded tree paths are considered their test subsessions. When a new test has to be merged with the CTS, the algorithm should evaluate all possible paths in the ECT. In order to keep track of the available tree paths: and avoid the complexity of the generalized tree travel problem, a list of potentially expandable test paths (ETP) is kept. This list is kept by means of special nodes that are inserted as leaf nodes within each ETP of ECT. These leaf nodes are called gaps and are depicted as hatched or shaded nodes in figures 1 and 2. There are two types of gaps. The first set of gaps (hatched) are those "rest gaps" left behind each merging step, like it was the case of \( GAP_1 \).
Table 1: Comparison of Power-Test Characteristics for the PTS Approaches (First Example)

<table>
<thead>
<tr>
<th>Power Constraints</th>
<th>TL</th>
<th>MPD</th>
<th>AVPD</th>
<th>PDD</th>
<th>RMS</th>
<th>TL</th>
<th>MPD</th>
<th>AVPD</th>
<th>PDD</th>
<th>RMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>103</td>
<td>202</td>
<td>131</td>
<td>88</td>
<td>67.5</td>
<td>103</td>
<td>87</td>
<td>24.7</td>
<td>66.5</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>103</td>
<td>181</td>
<td>126</td>
<td>88</td>
<td>67.5</td>
<td>103</td>
<td>87</td>
<td>24.7</td>
<td>66.5</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>95</td>
<td>164</td>
<td>118</td>
<td>88</td>
<td>67.5</td>
<td>103</td>
<td>87</td>
<td>24.7</td>
<td>66.5</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>95</td>
<td>151</td>
<td>108</td>
<td>88</td>
<td>67.5</td>
<td>103</td>
<td>87</td>
<td>24.7</td>
<td>66.5</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>95</td>
<td>140</td>
<td>100</td>
<td>88</td>
<td>67.5</td>
<td>103</td>
<td>87</td>
<td>24.7</td>
<td>66.5</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>95</td>
<td>130</td>
<td>92</td>
<td>88</td>
<td>67.5</td>
<td>103</td>
<td>87</td>
<td>24.7</td>
<td>66.5</td>
<td></td>
</tr>
</tbody>
</table>

and $GAP_i - t_k$ in the above example. They are similar to the uncomplete branches of the binary tree from [3]. The second set of gaps (shaded), are actually bogus gaps generated as the superposition of the leaf nodes and their twins as in the equivalence given at the right of figure 2. They are generated in order to keep track of "non-saturated" tree paths, which are also potential ETP's. By "non-saturated" tree path is meant any ETP who's accumulated power dissipation is still under the given power dissipation limit.

Figure 3: Power-Test Scheduling Chart

3.2. ADAPTED SCHEDULING APPROACH

A clear parallel between the HLS scheduling problem and the PTS scheduling problem is given by the similarities between the c-steps in HLS and the test (sub)sessions in PTS, between operations (HLS) and block-tests (PTS), and between hardware resource constraints (HLS) and power dissipation constraints (PTS). Therefore, there is an obvious coincidence between the process of assigning operations to c-steps (HLS scheduling) and the process of assigning block-tests to test (sub)sessions (PTS).

A comparison of the classical HLS approaches adapted to the power-constrained block-test scheduling problem is given in [4]. Firstly, a classical HLS register allocation algorithm such as the left-edge algorithm (HLS-LEA) was adapted there. The approach was named power-test scheduling based on left-edge algorithm (PTS-LEA). Secondly, the HLS list scheduling algorithm (HLS-LS) was employed as a greedy power-test list scheduling algorithm (PTS-LS). In PTS-LS, the next test session expansion was carried out using a local priority function. The local priority function was used to generate solutions which are as close as possible to the global solution. The local priority function was given by a system of two lists. Firstly, the list of block-tests left at a certain moment to be scheduled, which are ordered by the block-test mobility. Secondly, the list of test (sub)sessions to be expanded ordered by their accumulated power dissipation. There are two global priority functions. The first one is given by a system of two lists. Firstly, the list of block-tests left at a certain moment to be scheduled, which are ordered by the block-test mobility. Secondly, the list of test (sub)sessions to be expanded ordered by their accumulated power dissipation. Local priority functions do not render all the time optimal solutions. Therefore, global priority functions are preferable. The main difference between list scheduling (LS) and force-directed scheduling (FDS) approaches is the forecasting ability of their priority functions. Thus, a Force function was employed in PTS-FDS to steer the test scheduling so that the final solution has a more balanced and efficient test power-dissipation. Finally, the mean-square-error based scheduling (PTS-MSE) algorithm aimed to achieve a balanced outcome merely by assessing the power-dissipation distribution graphs (PCDG) and the effect of block-test/test-subsession assignments by using a least square error (MSE) function. Unlike the PTS-FDS approach, the time consuming stage of Force calculations was avoided by using the MSE function, resulting in a computationally more efficient solution.

It has been noticed in [4] that the PTS-LS-like approaches (PTS-LEA and PTS-LS) generate rapidly results which exhibit usually very good test application times. The lack of their solution is that their power dissipation characteristics are poor. That is because the power dissipation distribution is not balanced. These characteristics can be improved though by the PTS-FDS-like approaches (PTS-FDS and PTS-MSE). PTS-FDS-like approaches consume more computational time on calculating the global priority function but render solutions without power spikes. This is the goal of the two-step PTS-FDS scheduling approach proposed here. Firstly, to generate a solution exhibiting a good test application time using PTS-LS-like approaches. And secondly, to balance the power dissipation distribution of this solution by running on it PTS-FDS-like approaches. Because of space reasons the authors chose for this paper only the PTS-LS approach to accomplish the first step and the PTS-FDS approach for the second step.

The test application time of a power-test scheduling solution is given by the sum of the test lengths given by growing tree's roots. In order to keep this time characteristic unchanged, the PTS-FDS algorithm starts re-shuffling the block-tests leaving the root block-
MPD is the maximum power dissipation over the final power-test. If the PTS-LS approach is represented as a graph, the total test application time of the test scheduling solution is given by the Pareto-front (PDD), and root mean square power dissipation (RMS). TL represents the ideal power dissipation balance over the test application time.

4. EXPERIMENTAL RESULTS

In this section, two test scheduling examples are presented. The first example is discussed in order to provide a deeper insight into the results of this approach. It is the second example from a previous study. Thus, a comparison of the power-characteristics exhibited by power-test scheduling solutions is discussed for a block-test set example taken from the ASIC Z design discussed earlier in [1] and [2].

The power-test scheduling chart solution of this approach for the first example from [4] without power constraints is depicted in Figure 3. The power-characteristics of the chart are also given there: test length (TL), maximum power dissipation (MPD), average power dissipation (AVPD), total power dissipation dispersion (PDD), and root mean square power dissipation (RMS). TL represents the power-test scheduling chart. It is calculated as the ratio between the time required to complete the test application and the total time required for the test.

Table 2: Comparison of Power-Test Scheduling Characteristics for the PTS Approaches (Second Example)

<table>
<thead>
<tr>
<th>Power constraints</th>
<th>PTS-LS scheduling</th>
<th>PTS-FDS scheduling</th>
<th>PTS-LSFDS scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>TL</td>
<td>MPD</td>
<td>AVPD</td>
<td>PDD</td>
</tr>
<tr>
<td>800</td>
<td>441</td>
<td>597</td>
<td>228.18</td>
</tr>
<tr>
<td>700</td>
<td>441</td>
<td>597</td>
<td>228.18</td>
</tr>
<tr>
<td>600</td>
<td>441</td>
<td>597</td>
<td>228.18</td>
</tr>
<tr>
<td>500</td>
<td>441</td>
<td>597</td>
<td>228.18</td>
</tr>
<tr>
<td>400</td>
<td>441</td>
<td>597</td>
<td>228.18</td>
</tr>
<tr>
<td>300</td>
<td>441</td>
<td>597</td>
<td>228.18</td>
</tr>
<tr>
<td>200</td>
<td>441</td>
<td>597</td>
<td>228.18</td>
</tr>
</tbody>
</table>

Table 2: Comparison of Power-Test Scheduling Characteristics for the PTS Approaches (Second Example)

Tests (test session set) are unchanged. Then, by using the force-directed approach, the block-tests are re-accommodated. In the growing tree generating new test subsessions. The complexity of the PTS-LS approach is $O(n^3)$, while the complexity of the PTS-FDS approach is $O(n^3)$. Since this approach runs PTS-LS and PTS-FDS sequentially, its complexity is $O(n^3)$.

5. CONCLUSIONS

This power-constrained block-test scheduling approach is based on the classical LS and FDS algorithms adapted to an extended tree generating technique. This two-step approach tends to improve the power characteristics of PTS-LS solutions for loose power constraints. Generally, it generates power-test scheduling solutions exhibiting better power characteristics than both PTS-LS and PTS-FDS approaches. Thus, this approach is mostly beneficial when the test application time of the PTS-LS solutions is shorter than that of the PTS-FDS solutions. This positive result would trade-off the longer run time of this approach, which is roughly the sum of PTS-LS and PTS-FDS approaches' run times.

6. REFERENCES