Distribution-Graph Based Approach and Extended Tree Growing Technique in Power-Constrained Block-Test Scheduling

Valentin Mureşan, Xiaojun Wang
Dublin City University, Ireland
muresanv@eeng.dcu.ie

Valentina Mureşan, Mircea Vlăduţiu
"Politehnica" University of Timişoara, România
vmuresan@cs.utt.ro

Abstract

A distribution-graph based scheduling algorithm is proposed together with an extended tree growing technique to deal with the problem of unequal-length block-test scheduling under power dissipation constraints. The extended tree growing technique is used in combination with the classical scheduling approach in order to improve the test concurrency having assigned power dissipation limits. Its goal is to achieve a balanced test power dissipation by employing a least mean square error function. The least mean square error function is a distribution-graph based global priority function. Test scheduling examples and experiments highlight in the end the efficiency of this approach towards a system-level test scheduling algorithm.

1 INTRODUCTION

As the VLSI device technologies become mature, and larger and denser memory ICs are implemented for high-performance digital systems, the *power dissipation* becomes a critical factor and can no longer be ignored either in normal operation of the system or under *testing conditions*. Moreover, VLSI circuits running in test mode may consume more power than when running in normal mode [1]. Thus, the heat dissipated during test application is already influencing the test design methodology for practical circuits (e.g., MCMs). A solution towards alleviating this problem is the *test scheduling*. Test scheduling is strongly related to test concurrency. *Test concurrency* is a design property which strongly 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 which lacks of practical solutions. An efficient scheme for overlaying the block-tests, called *extended tree growing technique*, is proposed together with a *distribution-graph based scheduling* algorithm to search for power-constrained block-test scheduling profiles in a *polynomial time*. This approach exploits test parallelism under power constraints. This is achieved by overlaying the block-test intervals of compatible sub-circuits 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.The test scheduling discipline assumed here is the *partitioned testing with run to completion* defined in [2]. A *constant additive model* is employed for power dissipation analysis and estimation throughout the algorithm.

2 PROBLEM FORMULATION

The components which are required to perform a test are known as *test resources* and they may be shared among the blocks under test. 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. Depending on the test design methodology selected, once a *resource set* is compiled for each block-test t_i, 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*.

If $p(t_i)$ is the instantaneous power dissipation during test t_i and $p(t_j)$ is the instantaneous power dissipation during test t_j, then the power dissipation of a test session consisting of just these two tests is approximately $p(t_i) + p(t_j)$. 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 here for power estimation. That is, a constant power dissipation value $P(t_i)$ is associated with each block test t_i.

1081-7735/00 $10.00 © 2000 IEEE

For high-level approaches the power dissipation $P(t_i)$ of a test t_i could be estimated in three ways: *average power dissipation*, *maximum power dissipation* and, *RMS power dissipation* over all test steps in t_i. 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 $P(s_j)$ for a test session s_j can be defined as: $P(s_j) = \sum_{t_i \in s_j} P(t_i)$, while the power constraint in test scheduling is defined as: $P(s_j) \leq P_{max}\ \forall j$.

3 PROPOSED APPROACH

Power dissipation during test was seldom under research so far. A few 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. Unfortunately, the above approaches are not efficient at high levels, where the problem becomes NP-complete for big VLSI devices. The BIST scheduling approach given in [1] was one of the first to take into account power dissipation during test scheduling. It performs global optimization considering also other factors such as block type, adjacency of blocks (device floor plan). But in complex VLSI circuit designs, the block-test set is huge (especially at RT level) and ranges in test lengths. Therefore, this approach focuses only on problem's definition and analysis, whereas the proposed approach is not time efficient for huge block test sets.

[3] carried out for the first time an analysis of this problem at IC level, but it is only a theoretical one. It is, basically, a compatible test clustering, where the compatibility among tests is given by test resource and power dissipation conflicts at the same time. From an implementation point of view the identification of all cliques in the graph of compatible block tests belongs to the class of NP-complete problems. Instead, a distribution-graph based scheduling algorithm is proposed in this paper together with the extended tree growing technique to deal with the power-constrained test scheduling problem. The proposed approach belongs to the so called *unequal-length block-test scheduling* class, because it deals with tests for blocks of logic, which do not have equal test lengths. It has a polynomial complexity, which is very important for the efficiency of the system-level test scheduling. The algorithm is meant to be part of a system-level block-test approach to be applied on a modular view of a test hierarchy. The elements of this hierarchy could be given at any of the high-level synthesis domains, between the system and RT levels: subsystems, backplanes, boards, MCM's, IC's (dies), macro blocks and RTL transfer blocks. The lowest level block the test hierarchy accepts is the RTL one, but at this level it is assumed that a test-step level scheduling has already been taken into consideration and applied. Generally speaking any node in the hierarchy (apart from leaves) has different subnodes as children. Every test node t_i is characterized by a few parameters. These parameters are determined after test scheduling optimization has been performed on the node. They are given for every test node t_i: test application time T_i, power dissipation P_i, and test resource set $RES.SET_i$. This approach assumes a bottom-up traversing of the test hierarchy using a *divide et impera* optimization style. At a certain moment, the subnodes of a certain node are considered for optimization in order to get an optimal or near optimal sequencing or overlaying of them complying with the power constraints.

3.1 TREE GROWING APPROACH

Due to the wide range of test lengths exhibited by the block-test set applied to a complex VLSI circuit, it is possible to schedule some short tests to begin when subcircuits with shorter testing time have finished testing, while other subcircuits with longer testing time have not (if they are compatible). The *tree growing technique* given in [4] 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 is achieved by means of a *binary tree structure* (not necessarily complete), called *compatibility tree*, which was based on the compatibility relations amongst the tests.

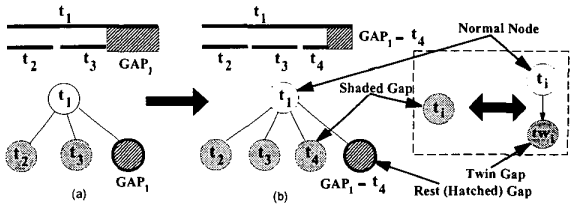


Figure 1. Merging Step Example

Nevertheless, a big drawback in [4] 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 example depicted in figure 1. Therefore an *expanded compatibility tree* (ECT), given by means of a *generalized tree*, is proposed here to overcome this problem. The sequence of nodes contained in the same tree path of an ECT 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 the CTS, such that every test contained in the nodes of P is compatible to t. The compatibility relationship 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 decreasing from root to leaf. Thirdly, the power dissipation accumulated on the above tree path should be less than or equal to P_{max}.

A *merging step* example is given in figure 1. The partial test schedule chart is given at the top, while the partially grown compatibility tree is given at the bottom. Let us assume that tests t_2, t_3 and t_4 are compatible to t_1, while they are not compatible to each other. Again let us assume that 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 $T_2 + T_3 < T_1$. Finally, let us assume that a new test t_4 has to be scheduled in the partial test schedule depicted in figure 1(a). As can be seen, there is a gap GAP_1 given by the following test length difference: $GAP_1 = T_1 - (T_2 + T_3)$. Thus, a merging step can be achieved, if $T_4 \leq GAP_1$, 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 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 avail of all possible paths in the ECT. In order to keep track of the available tree paths and to avoid the complexity of the generalized tree travel problem, a list of potentially *expandable tree paths* (ETP) is kept. This list is kept by means of special nodes that are inserted as leaf nodes in each ETP of the expanded compatibility tree. These leaf nodes are called *gaps* and are depicted as hatched or shaded nodes in figure 1. There are two types of gaps. The first set of gaps (hatched) are those "rest gaps" left behind each merging step, as in the case of GAP_1 and $GAP_1 - t_4$ in the above example. They are similar to the incomplete branches of the binary tree from [4]. The second set of gaps (shaded), are actually bogus gaps generated as the superposition of the leaf nodes and their twins as in the right equivalence given in figure 1. They are generated in order to keep track of "non-saturated" tree paths, which are also potential ETPs. By "non-saturated" tree path is meant any ETP with accumulated power dissipation still under the given power dissipation limit.

3.2 DISTRIBUTION-GRAPH-BASED SCHEDULING APPROACH

A clear parallel between the HLS scheduling problem and the power-constrained test scheduling (PTS) problem is given by the similarities between the c-steps in HLS and the test sessions (subsessions) 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). With the tree growing technique, proven efficient HLS algorithms can be easily adapted to the PTS problem modeled as an extended tree growing process. A comparison of the PTS approaches is given in [5]. The HLS list scheduling algorithm (HLS-LS) was employed as a greedy power-test list scheduling algorithm (PTS-LS). In the PTS-LS algorithm the block-tests are initially ordered by their *test mobility* before being scheduled. The test mobility TM_i of a block-test t_i is inversely proportional to the product of its test length T_i and its power dissipation P_i: $TM_i = \frac{1}{T_i * P_i}$. The sorted block-tests are then iteratively scheduled into the available test (sub)sessions (ETPs). When the power dissipation is exceeded the block-tests to be currently scheduled are deferred to the other test (sub)sessions (ETP) left for further expansion. In PTS-LS, the next test (sub)session expansion was carried out using a *local priority function*. Local priority functions do not render all the time optimal solutions. Therefore, global priority functions are preferable. The main difference between the list scheduling (LS) approach and the force-directed scheduling (FDS) approach is the forecasting ability of their priority functions. The FDS approach uses a *global priority function* called *Self Force*. The *Self Force* function is employed to steer the block-tests' assignments to test (sub)sessions. The selection of the test subsession in which the selected block-test will be placed is based on achieving in each test subsession a balanced distribution of block-tests' parameters, i.e. power dissipation and test concurrency. This is achieved using three steps: determination of block-tests' time frames, creation of power-test distribution graphs and, calculation of *Self Forces*.

The *distribution graph concept* based scheduling (PTS-MSE) algorithm given here is a parallel to the HLS scheduling algorithm given in [6]. It aims at achieving a balanced outcome merely by assessing the *power-concurrency distribution graph* (PCDG's) and the effect of block-test/test-subsession assignments by using a least *mean square error* (MSE) function. Unlike the PTS-FDS approach given in [5], the time consuming *Self Force* stage calculations are avoided here by using the MSE function approach, resulting in a computationally more efficient solution. This is achieved using the only two steps summarized below:

Determination of time frames: the first step consists of determining the time frames of each block-test by evaluating the set of test subsessions (ETPs) where the block-test can be merged. The ETPs expandable at a certain moment with a block-test do not have to be adjacent and, therefore, a block-test's time frame in PTS-MSE, and also in PTS-FDS, is not or does not have to be contiguous. This is the outstanding difference between the HLS-FDS approach and PTS-MSE (PTS-FDS) approaches. In HLS-FDS the uniform probability of an operation to be assigned to a c-step

is taken into consideration in order to achieve a balanced operation concurrency. On the other hand, the goal of PTS-MSE is to balance mainly the power dissipation and, indirectly, the test concurrency, while keeping tight the test application time as much as possible. Therefore, a *power dissipation probability* is to be used here instead and was defined for the first time in [5] for the PTS-FDS approach.

Creation of distribution graphs: the next step is to take the sum of the block-tests' probabilities for each ETP (gap) of the partial test schedule and add them on top of the power dissipation accumulated so far in the partial power-test chart. The resulting *power-concurrency distribution graph* (PCDG) indicates the power dissipation expectations and indirectly the possible test concurrency distribution of the future test schedule. The PCDG's formula is given in [5].

Since the order of block-test assignment affects the scheduling process, the following approach is adapted to obtain optimum results. First, all block-tests are ordered by their mobility. High mobility block-tests have little effect on the overall block-test's power and concurrency distribution because of their long time frames which result in low probability values. This means that it is unnecessary to schedule high mobility block-tests early. This list is used to schedule all block-tests one by one without the need to reorder the list, thus preserving the algorithm's low complexity.

The PSEUDOCODE of the PTS-MSE Algorithm:
-initialize the $GrowingTree$**, the** $BlockTestList$ **and the** $GapsList$**;**
-initialize and sort all block-tests according to block-test mobility, test length and power consumption;
-while there are unscheduled block-tests do
/* $BlockTestList$ **is not empty*/ {**

- **take the next block-test out from the sorted list;**
- **for each test subsession into which the block-test could be scheduled do {**
  - **assign the block-test tentatively to the test subsession;**
  - **update time frames of incompatible block-tests;**
  - **calculate distribution graph for the modified growing tree;**
  - **evaluate the mean square error (MSE) function;**
- **end for;}**
- **schedule block-test into the test subsession for which the lowest MSE value was found;**
- **update time frames of incompatible block-tests;**
- **update distribution graph;**

-end while;}

In order to optimize the power-dissipation (test-concurrency) throughout the test application, it is necessary to balance the overall PCDG. Though, assigning a block-test to a specific gap (expandable test subsession) often affects the time frames of the other "ready" block-tests, which may become *incompatible* to the test (sub)sessions newly generated after a block-test is scheduled in the current test (sub)session. Thus, scheduling a particular block-test t_j into a certain test subsession ts_i affects time frames of other block-tests. As a result, probability values of these block-tests vary and modified distribution graphs $PCDG'_j$ should be determined for each $t_j \rightarrow ts_i$ assignment. To investigate the effect of different test subsession assignments on the block-tests' distribution, the balance of the temporary $PCDG'_j$ is assessed knowing that a good schedule has a balanced $PCDG$. The difference between the $PCDG$ values and an average value (AVG) provides an indication of the graph balance. The average value is obtained from the original $PCDG$ using:

$$AVG = \frac{1}{N_{TS}} \sum_{i=0}^{N_{TS}-1} PCDG(i), \quad (1)$$

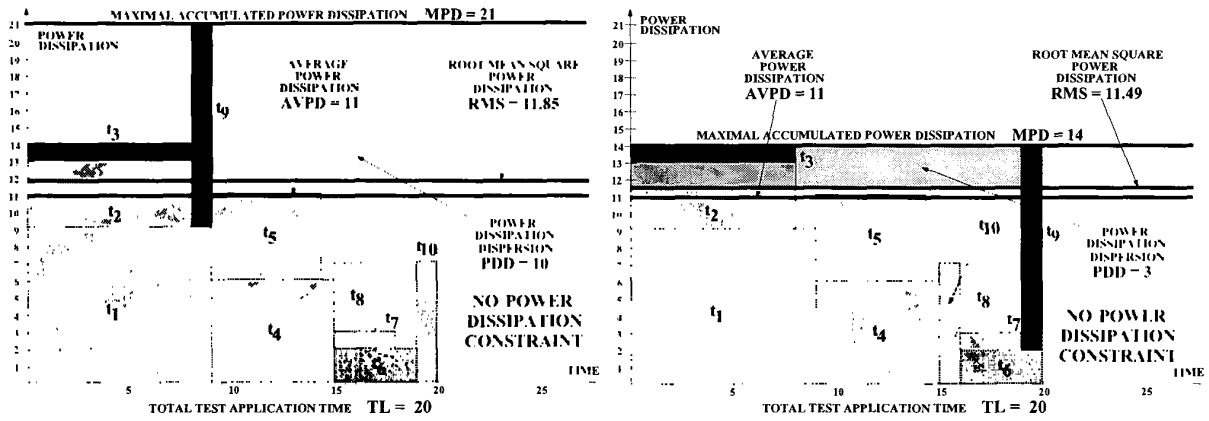
where N_{TS} is the number of test subsessions in the schedule. The differences in the $PCDG$ are used to obtain a numerical value for the schedule quality using a *mean square error (MSE)* function:

$$MSE(i) = \frac{1}{N_{TS}} \sqrt{\sum_{i=0}^{N_{TS}-1} (PCDG'_j(i) - AVG)^2}, \quad (2)$$

where $PCDG'_j(i)$ is the modified power-concurrency distribution graph for a $t_j \rightarrow ts_i$ assignment. Having determined the MSE values for all valid test subsessions ts_i, the block-test t_j is finally scheduled into the test subsession which results in the lowest MSE value. This is followed by adjusting the time frames of *incompatible block-tests* and updating the $PCDG$ values. This procedure is repeated until all block-tests are scheduled. The pseudocode is given above. The complexity of the PTS-MSE algorithm can be derived in the following way. Firstly, each iteration of the algorithm schedules one block-test. This implies there are n iterations (n is the number of block-tests). Secondly, within each iteration, for a block-test to be scheduled there are at most n test subsession (gaps) for which $PCDG$ must be calculated. Finally, for each tentative block-test to test subsession assignment, there may be at most *n-1* block-tests incompatible to the current one to be affected, and their incompatibility force must also be calculated. This assumption is a very conservative upper bound. The combined effect of the above three considerations yields the combined $O(n^3)$ complexity.

4 EXPERIMENTAL RESULTS

In this section three test scheduling examples are presented. The first one is a small example meant to give an



(a) List Scheduling Solution

(b) Distribution Graph Based Scheduling Solution

Figure 2. Power-Test Scheduling Charts (No Power Limit)

idea about the type of results generated by the PTS-MSE algorithm. Then a bigger second example, chosen randomly, is discussed in order to provide a deeper insight into the results of the PTS-MSE algorithm. In the end the PTS-MSE algorithm is compared in terms of "complexity vs results" for a block-test set example taken from [5]. This example is developed on the ASIC Z design proposed earlier on in [1].

Suppose for the first example the following block-test set example. Their parameters are specified in the order: power consumption, test length and their compatibility list. For simplicity reasons, the block-tests are already ordered by test length and power consumption keys. The power-test scheduling charts given by the PTS-LS and PTS-MSE algorithms for this block-test set are depicted in figure 2. It can be seen in figure 2(b), that the PTS-MSE approach gives a power-test chart solution exhibiting a more balanced power dissipation than the PTS-LS solution given in figure 2(a).

t_1 (9, 9, $\{t_2, t_3, t_5, t_6, t_8, t_9\}$)
t_2 (4, 8, $\{t_1, t_3, t_7, t_8\}$)
t_3 (1, 8, $\{t_1, t_2, t_4, t_7, t_9, t_{10}\}$)
t_4 (6, 6, $\{t_3, t_5, t_7, t_8\}$)
t_5 (5, 5, $\{t_1, t_4, t_9, t_{10}\}$)
t_6 (2, 4, $\{t_1, t_7, t_8, t_9\}$)
t_7 (1, 3, $\{t_2, t_3, t_4, t_6, t_8, t_9\}$)
t_8 (4, 2, $\{t_1, t_2, t_4, t_6, t_7, t_9, t_{10}\}$)
t_9 (12, 1, $\{t_1, t_3, t_5, t_6, t_7, t_8, t_{10}\}$)
t_{10}(7, 1, $\{t_3, t_5, t_8, t_9\}$)

The power-test characteristics of the power-test scheduling charts are given in figure 2: test length (TL), maximum power dissipation (MPD), average power dissipation (AVPD), power dissipation dispersion (PDD), and root-mean-square power dissipation (RMS). TL represents the total test application time of the test scheduling solution. MPD is the maximum power dissipation over the final power-test scheduling solution. AVPD is considered the ideal MPD when all the ETPs would exhibit the same accumulated power dissipation, that is, the power dissipation would be fully balanced over the power-test scheduling chart. AVPD is calculated as the power-test area, taken up by the chart, divided by TL. The rectangle given by AVPD and TL would be the ideal power-test scheduling chart and, therefore, the ideal test scheduling profile. PDD is directly proportional to the accumulated power dissipation dispersion over the power-test scheduling chart, which is considered to be given by the power-test area left unused inside the power-test rectangle having MPD and TL as sides. PDD is calculated as the difference between MPD and AVPD. RMS gives the root mean square value for the power dissipation distribution of a scheduling chart.

Secondly, the PTS-MSE algorithm is experimented for a 50 block-tests set chosen randomly, where the degree of resource compatibility between the block-tests is high (around 90%). The degree of resource compatibility between the block tests gives the dimension of the solution space. The higher the resource compatibility degree, the larger the solution space. This test scheduling example is run in order to draw the characteristics of the solutions selected by the PTS-MSE approach from a bigger solution space. In figure 3 these characteristics are generated for a range of power dissipation constraints from totally relaxed to fully tight. It can be seen there that TL grows almost linearly with the power constraint increase, whereas MPD and PDD exhibit a linear decrease. In [5] comparing the

PTS-LS (PTS-LEA) scheduling solutions on the one hand with the PTS-MSE (PTS-FDS) results on the other hand, it has been noticed that the former gave "noisier" solutions for relaxed constraints. That is, the PTS-LS characteristics (TL, MPD, AVPD, PDD, RMS) do not have a smooth trend while the power constraints are ranged from relaxed to tight. Intuitively, the global priority function helps the PTS-MSE (PTS-FDS) approaches to have a global view over the solution space and to pick up most of the times better solutions. Overall, the PTS-MSE (PTS-FDS) approaches give power-test scheduling profiles which exhibit a more balanced power distribution as was the case in the example from figure 2. That is, the solutions given by PTS-MSE (PTS-FDS) algorithms have smoother TL and AVPD characteristics, and smaller PDD and MPD characteristics. Though, PTS-MSE (PTS-FDS) algorithms are computationally more expensive than the PTS-LS (PTS-LEA) approaches. Moreover, while the PTS-FDS approach is computationally slightly more expensive than the PTS-MSE approach, the solutions given by the former are slightly more balanced.

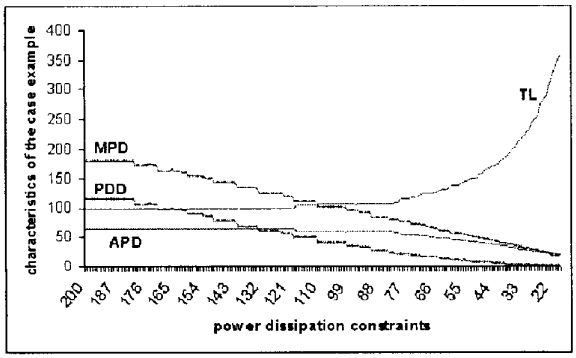


Figure 3. PTS-MSE Solution Characteristics I

For the third example a practical testbench is considered. An extended case [5] of the ASIC Z design given in [1] is experimented with the PTS-MSE approach. The testbench has 27 tests spread over 9 cores. The results are given in table 1 over a range of the power constraint. They exhibit the same features as the results of the second example. Unfortunately, the results of the experiments run here cannot be compared to the ones given for the ASIC Z case in [1, 3]. That is due to the fact that the test scheduling discipline assumed in [1, 3] is the *nonpartitioned testing* defined in [2], whereas the one assumed in this paper is the *partitioned testing with run to completion*. The nonpartitioned testing assumes that no tests can be started until all tests in the previous session is completed, which is opposite the case of this paper.

power constraints	PTS-MSE scheduling solutions for the ASIC Z example				
	TL	MPD	AVPD	PDD	RMS
900	320	465	314.46	150.54	348.81
800	320	465	314.46	150.54	348.81
700	320	465	314.46	150.54	348.81
600	320	464	314.16	149.54	352.19
500	320	464	314.46	149.54	352.19
400	360	387	279.52	107.48	286.53
300	400	296	251.57	44.44	256.76
200	670	176	150.19	25.81	151.04

Table 1. PTS-MSE Solution Characteristics II

5 CONCLUSIONS

The work proposed in this paper has been carried out based on the ascertained fact that not a lot approaches to tackle the power-constrained test scheduling problem have been identified so far. A classical distribution graph based scheduling algorithm is proposed for the power-constrained test scheduling problem, which is modeled as a growing tree. Thus, by means of the tree growing approach, classical algorithms are re-used to provide fast results. Its polynomial complexity is beneficial to the system-level test scheduling problem. Even though it does not guarantee optimal block-test scheduling solutions, its fast final results can be used as starting points by near-optimal block-test scheduling approaches (e.g. simulated annealing, tabu search) to get an improved solution.

References

[1] Y. ZORIAN: **A Distributed BIST Control Scheme for Complex VLSI Devices** - *Proceedings of The 11th IEEE VLSI Test Symposium*, pp. 4-9, Apr, 1993.

[2] G.L. CRAIG, C.R. KIME, K.K. SALUJA: **Test Scheduling and Control for VLSI Built-In Self-Test** - *IEEE Transactions on Computer*, Vol. 37, No. 9, pp. 1099–1109, Sep, 1988.

[3] R.M. CHOU, K.K. SALUJA, V.D. AGRAWAL: **Scheduling Tests for VLSI Systems Under Power Constraints** - *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 5, No. 2, pp. 175–185, Jun, 1997.

[4] W.B. JONE, C. PAPACHRISTOU, M. PEREIRA: **A Scheme for Overlaying Concurrent Testing of VLSI Circuits** - *Proceedings of the 26th Desing Automation Conference*, pp. 531-536, 1989.

[5] V. MURESAN, X. WANG, V. MURESAN, M. VLADUTIU: **A Comparison of Classical Scheduling Approaches in Power-Constrained Block-Test Scheduling** - *IEEE Test Conference (ITC) 2000*, accepted paper.

[6] P. KOLLIG, B.M. AL-HASHIMI, K.M. ABBOTT: **Efficient Scheduling of Behavioral Descriptions in High-Level Synthesis** - *IEE Proceedings-Computers And Digital Techniques*, Vol. 144, No. 2, pp. 75–82, Mar, 1997.