# A Comparison of Classical Scheduling Approaches 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

*Classical scheduling approaches are applied here to overcome the problem of unequal-length block-test scheduling under power dissipation constraints. List scheduling-like approaches are proposed first as greedy algorithms to tackle the fore mentioned problem. Then, distribution-graph based approaches are described in order to achieve balanced test concurrency and test power dissipation. An extended tree growing technique is also used in combination with these classical approaches in order to improve the test concurrency having assigned power dissipation limits. A comparison between the results of the test scheduling experiments highlights the advantages and disadvantages of applying different classical scheduling algorithms to the power-constrained test scheduling problem.*

## 1 INTRODUCTION

Power dissipation has become a critical factor in the normal operation of nowadays' deep-submicron digital systems or under testing conditions. VLSI circuits running in test mode may consume more power than when running in normal mode. It is reported in [1] that one of the major considerations in test scheduling is the fact that heat dissipated during test application is typically significantly higher than the heat dissipated during the circuits' normal operation (sometimes 100 - 200% higher). *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.

*Power-constrained test scheduling* will soon become a very important issue for the SOC designs as well. Therefore, this paper focuses on the high-level power-constrained block-test scheduling problem which lacks of practical solutions. The test scheduling discipline assumed here is the *partitioned testing with run to completion* defined in [2]. An efficient scheme for overlaying the block-tests, called *extended tree growing technique* is employed to model the fore mentioned problem. Thus, traditional high-level synthesis approaches (e.g. left-edge algorithm, list scheduling and distribution-graph based scheduling) can be employed together with this model to search for power-constrained block-test schedule profiles in a *polynomial time*. The algorithm fully exploits test 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 does not go over the given limit. A *constant additive model* is employed for the power dissipation estimation throughout the algorithms.

## 2 PREVIOUS WORK

Power dissipation during test was seldom under research so far and focused at low levels. Approaches like [3, 4, 5] tackle the power dissipation problem during test application at gate-level. [3] improves the low correlation between consecutive test vectors generated by an ATPG. This low correlation greatly increases switching activity and thus the power dissipation. Scan-latch ordering and test-vector ordering techniques have been proposed in [4] to minimize the power dissipation at gate level in scan or combinational circuits. In [5] the authors present a test vector inhibiting technique to filter sets of consecutive non-detecting patterns of a pseudo-random test sequence generated by a LFSR.

Unfortunately, the above approaches are not efficient at high levels. 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 usually unknown at high-level. Moreover, in complex VLSI circuit designs, the block-test set is huge and ranges in test lengths. Thus, the work in [1] focused more on the


0-7803-6546-1/00 $10.00 

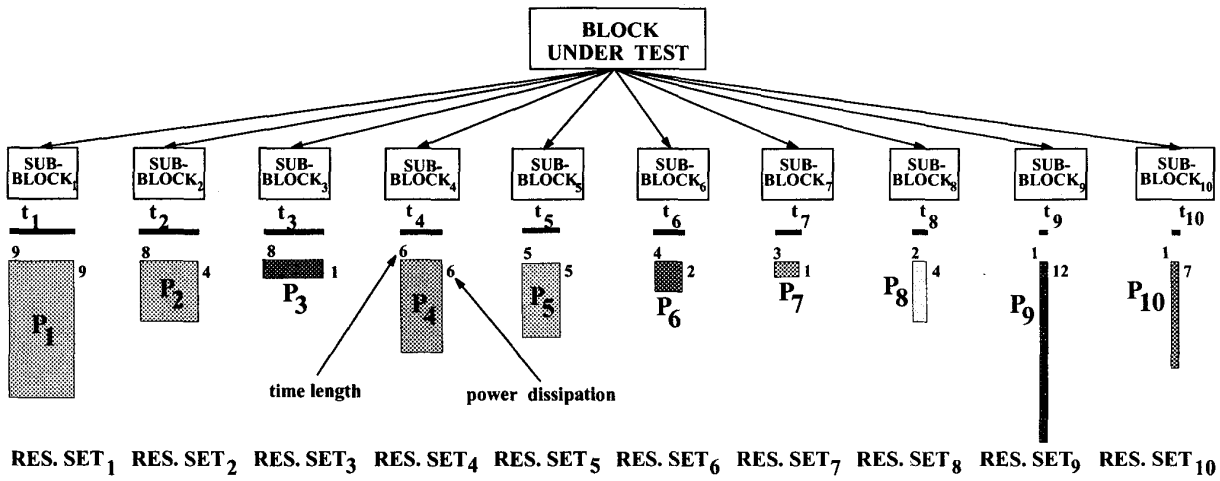



**Figure 1. Example of Block Under Test (Sub-Block Hierarchical View)**

problem definition than on proposing a polynomial complexity algorithm to solve it. [6] makes for the first time a thorough analysis of the power-constrained test scheduling (PTS) problem at IC level, but it is only a theoretical one. That is because it proposes a compatible test clustering technique which is an NP-complete approach. The compatibility among tests is given here by test resource and power dissipation conflicts at the same time. Recently, [7] proposed a greedy algorithm for the PTS problem, but only for the particular case of core-level test scheduling.

# 3 PROPOSED APPROACH

The proposed algorithm is an *unequal-length block-test scheduling* one because it deals with tests for blocks of logic, which do not have equal test lengths. 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 given at any of the high-level synthesis (HLS) 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 block in hierarchy (apart from leaves) has different sub-blocks as children. Every test block $t_i$ is characterized by a few parameters, which have been previously assigned to $t_i$, after the test scheduling optimization has been applied on the test block. These parameters are given in figure 1: 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 hierarchical test model within a *divide et impera* optimization style. Thus, at a certain moment the sub-blocks of a certain block are considered for optimization in order to get an optimal or near optimal sequencing or overlaying of them complying with the power dissipation constraints.

## 3.1 PROBLEM FORMULATION

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 the sum of the instantaneous powers of test $t_i$ and $t_j$. However, in reality the instantaneous power for each test vector is hard to obtain since it depends, e.g., in a CMOS circuit on the number of zero-to-one and one-to-zero transitions, which in turn could be dependent on the order of execution of test vectors. In order to simplify the analysis, a *constant additive model* is employed here for power estimation. A constant power dissipation value $P(t_i)$ is associated with each block test $t_i$. For high-level approaches the power dissipation $P(t_i)$ of a test $t_i$ could be estimated in three ways: *average power dissipation* over all test steps in $t_i$, *maximum power dissipation* (peak power) over all test steps in $t_i$ and, *RMS power dissipation*. The total power dissipation at a certain moment of the test schedule is computed by simply summing the power dissipation of the concurrently 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 usually defined as: $P(s_j) \leq P_{max}\ \forall j$.

### 3.2 TREE GROWING APPROACH

In complex VLSI circuit designs, the block-test set is huge and ranges in test lengths, especially at RT level. Thus, 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 [8] 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.

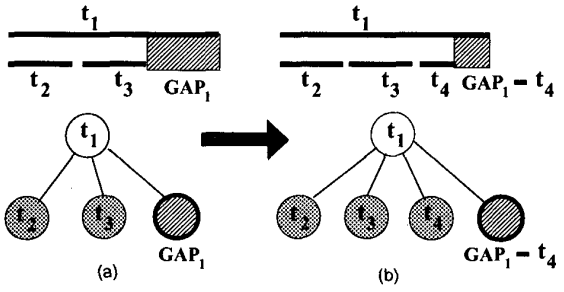



**Figure 2. Merging Step Example**

Nevertheless, a big drawback in [8] is that the compatibility tree is a binary one. This limits the number of children test blocks 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 2. Therefore an *expanded compatibility tree* (ECT), given by means of a *generalized tree*, is proposed here to overcome this problem. Figure 3 gives the final test scheduling chart and its ECT for the test scheduling example given in section 5. The power-test scheduling chart of the test scheduling chart given in figure 3 is depicted in figure 4(b). Figures 4(a) and 5(a) depict test scheduling charts for test scheduling solutions generated by the list scheduling-like approaches without and with ($P_{MAX} = 12$) power constraints. Figure 5(b) gives the power dissipation characteristics of the test scheduling chart from figure 5(a).

The sequence of nodes contained in the same tree path of an ECT represents an expansion of the CTS. Given a partial schedule 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 growing from leaf to root. 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 2. 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 2(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 2(b).

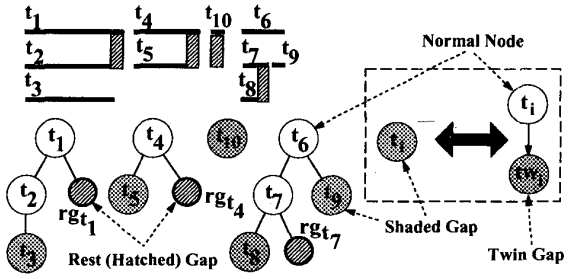



**Figure 3. Test Scheduling Chart and Extended Compatibility Tree Example**

The process of constructing CTSs 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 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 within each ETP of ECT. These leaf nodes are called *gaps* and are depicted as hatched or shaded nodes in figures 2 and 3. 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 [8]. 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 on the right hand side of figure 3. 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

who's accumulated power dissipation is still under the given power dissipation limit. The root nodes (test sessions) are considered by default "shaded" gaps before any test subsection is generated below them.

# 4 CLASSICAL HLS APPROACHES

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 (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). The biggest achievement of the tree growing technique is that proven efficient HLS algorithms can be easily applied to the power-constrained test scheduling problem modeled as an extended tree growing process.

## 4.1 LIST SCHEDULING APPROACHES

A classical HLS approach such as the left-edge algorithm proposed initially for the HLS register allocation problem is applied on power-constrained block-test scheduling in [9]. The approach, consisting of three algorithms, is named power-test scheduling based on left-edge algorithm (PTS-LEA). The HLS list scheduling algorithm is employed as a greedy power-test list scheduling algorithm (PTS-LS) in [10]. Both PTS approaches are list scheduling-like algorithms, the PTS-LS algorithm resembling a lot the first algorithm of the PTS-LEA approach.

In the PTS-LS-like approaches a local priority function, called *test mobility* $TM_i$, is defined for each block-test $t_i$ in order to sort initially the block-tests. Based on block-tests' parameters, this function has a test length $T_i$ component and a power dissipation $P_i$ component. Thus, in PTS-LS-like approaches, the block-tests are sorted in topological order by using the test length as primary key to order in descending order, and the power dissipation as secondary key, to order the block-tests having the same test length in descending order as well. The sorted block-tests are then iteratively scheduled in a greedy manner into the available test (sub)sessions (ETP). When the power dissipation is exceeded the block-tests to be currently scheduled are deferred for the other test (sub)sessions (ETP) left for further expansion. For space reasons the list scheduling-like approaches proposed to solve the PTS problem are not further detailed here. They are fully presented in [9] (PTS-LEA) and [10] (PTS-LS).

## 4.2 DISTRIBUTION-BASED APPROACHES

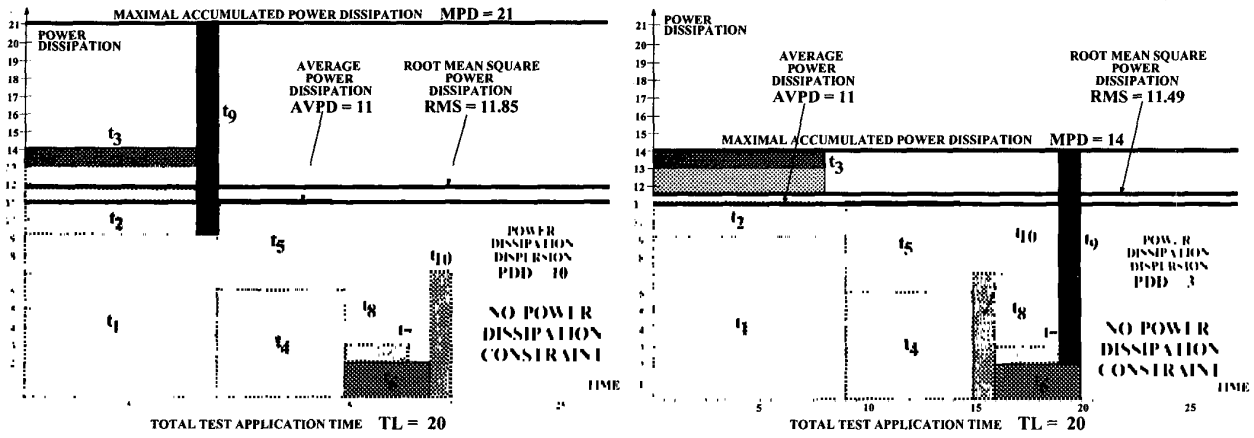
Local priority functions do not render all the time optimal solutions. Therefore, global priority functions are preferable. The main difference between the list scheduling-like approaches and the distribution-based approaches is the forecasting ability of their priority functions. A distribution-graph based priority function is employed in the latter approach to steer the scheduling so that the final solution is more efficient and has a more balanced power dissipation distribution.

### 4.2.1 FORCE-DIRECTED APPROACH

The intent of the HLS-FDS algorithm [11] is to reduce the number of functional units, registers and buses required, by *balancing the concurrency of the operations assigned to them*, but without lengthening the total execution time. Concurrency balancing helps to achieve high utilization - or low idle time - of structural units, which in turn minimizes the number of units required. This idea is borrowed here to implement a power-constrained test scheduling algorithm. The result is a power-constrained test scheduling based on FDS (PTS-FDS) approach. The objectives here are to achieve a test concurrency and power dissipation balance at the same time with test application time minimization, having given power dissipation limits. The set of test subsessions (ETPs) in the growing tree changes throughout the algorithm's execution because they are expanded and then changed with new test subsessions (hatched and shaded nodes). Consequently, the set of test subsessions is dynamic during the PTS-FDS approach, while their equivalent in the HLS version of FDS, the set of c-steps, is static. This is the first of the two differences that exist between the FDS approaches mentioned here.

The PTS-FDS algorithm is iterative, with one block-test scheduled at each iteration. The selection of the test subsession in which it will be placed is based on achieving in each test subsession a balanced distribution of power dissipation and test concurrency. This is achieved using the three step algorithm summarized below:

*Determination of time frames*: during the HLS-FDS approach, the first step consists of determining the time frames of each operation by evaluating the ASAP (as soon as possible) and ALAP (as last as possible) schedules. The time frames are contiguous in HLS-FDS. On the other hand, in the PTS-FDS approach the time frame of a block-test is the set of test subsessions (ETPs) where the block-test can be merged. At a certain moment, the ETPs expandable by a block-test do not have to be adjacent and, therefore, block-test's time frame in PTS-FDS is not or does not have to be contiguous. This is the second outstanding difference between the HLS-FDS and PTS-FDS approaches. In HLS-




(a) List Scheduling Solution

(b) Force-Directed Scheduling Solution

**Figure 4. Power-Test Scheduling Charts (No Power Limit)**

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. The goal of PTS-FDS 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. The power dissipation probability of a block-test $t_i$ to be assigned to a test subsession $ts_j$ is defined as the product of three components:

- the power dissipation $P_i$ of the block-test $t_i$;
- the compatibility probability between the current block-test $t_i$ and the rest of the block-tests assignable to the same test subsession $ts_j$. Its formula is:

$$Comp_{Prob}(t_i, ts_j) = 1 - \frac{C_{N_{incomp}}^{K-1}}{C_{N-1}^{K-1}}, \qquad (1)$$

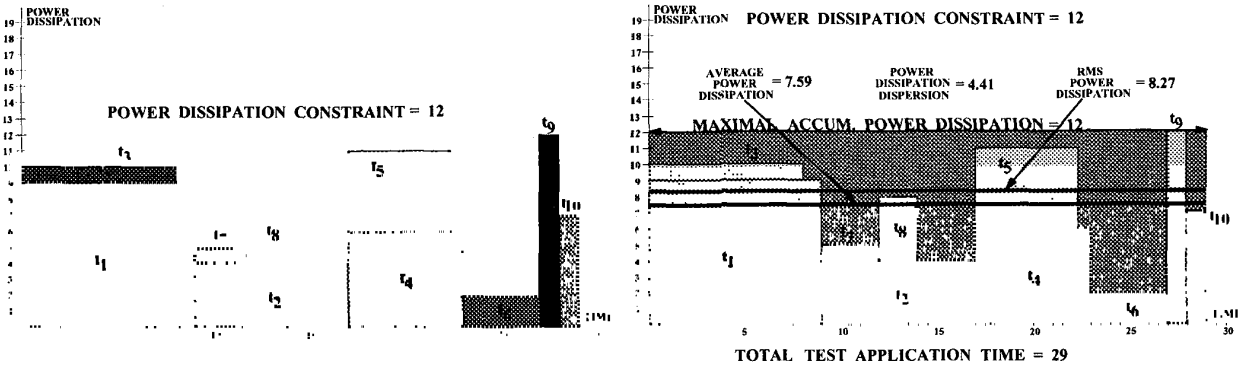
  where $N_{incomp}$ is the number of block-tests in $t_i$'s incompatibility list, $N$ is the total number of block-tests and $K$ is the number of block-tests compatible with test subsession $ts_j$. This probability is a measure of the chance to schedule block-test $t_i$ in test subsession $ts_j$ in parallel with the other $K$ block-tests;
- the uniform probability $Prob(t_i, ts_j)$ of assigning the current block-test $t_i$ to one of its time frame's test subsessions $ts_j$ to which it could be assigned. This probability is the same with the one employed in HLS-FDS.

*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 already in the partial power-test chart. The resulting *power-concurrency distribution graph* (PCDG's) indicates the power dissipation expectations and, indirectly, the possible test concurrency distribution of the future test scheduling solution. The PCDG's formula is:

$$PCDG(ts_j) = P_{ts_j} + \sum_{t_i} P_i * Comp_{Prob}(t_i, ts_j) * Prob(t_i, ts_j). \qquad (2)$$

*Calculation of Self Forces*: the final step is to calculate the force associated with every feasible test subsession (gap) assignment of the block-tests to be scheduled. For a given block-test $t_i$ the assignment force to test subsession $ts_j$ is given by $Force(j) = PCDG_j * x(i)$, where $PCDG_j$ is the current power-test distribution value in $ts_j$ and $x(i)$ is the change in the block-test's probability after assignment. The assignment is done by temporarily reducing the block-test's time frame to the selected test subsession. In PTS-FDS, the *Self Force* of a block-test is a quantity which reflects the effect of an attempted test subsession assignment on the overall power dissipation balance and, indirectly, on the test concurrency balance. This function is positive if the assignment causes an increase of power-test concurrency (power unbalance), and is negative for a decrease. The total *Self Force* associated with the assignment of a block-test $t_i$ to one of the test subsessions $ts_j$, where $j \in SETS$ (Set Of Expandable Test Subsessions), is:

$$Self\ Force(i) = \sum_{j \in SETS} Force(j). \qquad (3)$$




(a) List Scheduling Solution

(b) Power-Test Characteristics of the Solution

**Figure 5. Power-Test Scheduling Charts' Characteristics ($P_{max}$ = 12)**

### INCOMPATIBILITY FORCES

In order to optimize the power dissipation (test concurrency) throughout the test application, it is necessary to assign block-tests to test subsessions such that the power-dissipation/test-concurrency distribution values decrease and the overall PCDG is more balanced. Though, assigning a block-test to a specific gap (expandable test subsession) often affects the time frames of the rest of initially "ready" block-tests, which become incompatible to the test subsessions newly generated after assignment. This happens because scheduling a block-test is equivalent to reducing its time frame to one test subsession. This modification would propagate to the time frames of the other block-tests (initially assignable to the same test subsession) which are not assignable anymore to the just expanded test subsession.

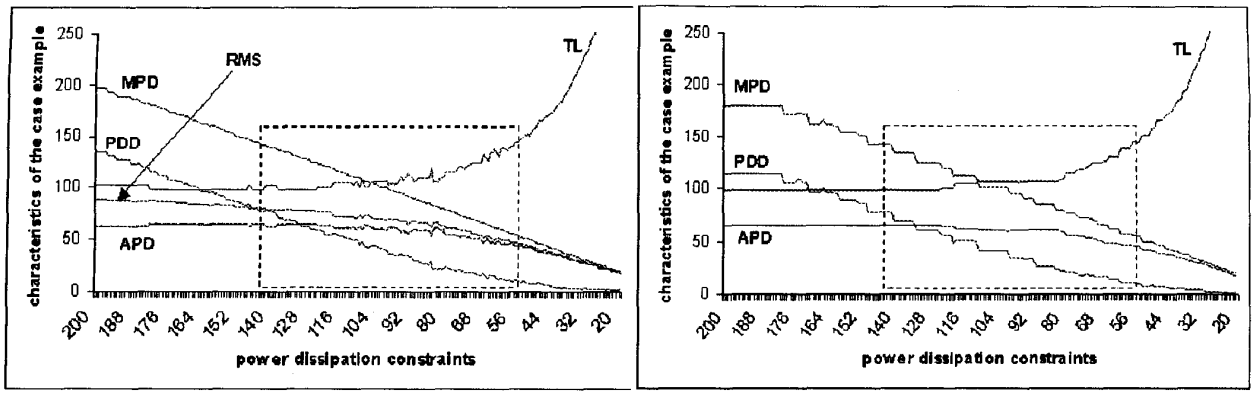
Thus, any block-test assignment usually creates additional forces that can reduce or even counter the originally intended improvement, so it is imperative that they be accounted for. Therefore, the force calculation must also be performed for all block-tests which became incompatible, but only when their time frame is affected. These forces are named in this approach *incompatibility forces* and they are calculated like the normal forces and added to the overall *Self Force*. After the calculation of the forces of all block-tests has been performed, the block-test with the lowest force or the best concurrency balancing is selected for test subsession assignment.

### PSEUDOCODE

The pseudocode of this approach is given in appendix A. The data structures used in it are: the $Growing\ Tree$ (GT) to model the ECT, $GapsList$ (GL) to model the list of potentially expandable gaps (shaded and hatched gaps), $BlockTestList$ (BTL) to keep the ordered but not yet merged block-tests. $CurTest$ is the block-test to be merged at a certain iteration. $CurGap$ is the gap under focus at a certain iteration to see whether it is expandable (compatible) with the $CurTest$. In the pseudocode "used" means that the block-test has already been merged in the ECT. $TwinGap$ is the newly generated shaded gap at every iteration and it will not be inserted in the $GapsList$ anymore after its generation, if its resulting compatibility list is null, i.e. it will not be an ETP. $RestGap$ is meant to keep the hatched gap generated at every iteration if it is not null, i.e. $CurTest$ covers completely $CurGap$. The SCHEDULE procedure in the pseudocode is similar to those employed in the other tree growing based approaches [9, 10].

### 4.2.2 MEAN-SQUARE-ERROR BASED APPROACH

The *mean-square-error* based scheduling (PTS-MSE) algorithm [12] aims to achieve 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, the time consuming stage of *Self Forces* calculations are avoided here by using the MSE function, resulting in a computationally efficient solution. This is achieved executing only the first two steps of the PTS-FDS in order to update the DGs. At each iteration, all the unscheduled block-tests are ordered by their *test mobility* and the block-test $t_j$ with the highest mobility is scheduled. The effect on the PCDG given by the assignment of $t_j$ to different test subsession $t_i$ is assessed knowing that a good scheduling solution has a balanced PCDG. The difference between the PCDG values and PCDG's average value (AVG) provides an indication of the graph balance. The average value is obtained using the following solution:




(a) Characteristics of the PTS-LS Results

(b) Characteristics of the PTS-FDS Results

**Figure 6. Power-Test Characteristics over the Range of Power Dissipation Constraints**

$$AVG = \sum_{i=0}^{N_{TS}-1} PCDG(i), \quad (4)$$

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:

$$MSE(j) = \frac{1}{N_{TS}} \sqrt{\sum_{i=0}^{N_{TS}-1} (PCDG'_j(i) - AVG)^2}, \quad (5)$$

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, 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 complexity of the PTS algorithms mentioned in this paper ranges from $O(n^2)$ for the list scheduling-like approaches to $O(n^3)$ for the distribution-based approaches. The PTS-MSE algorithm is computationally more efficient than the PTS-FDS algorithm because it eliminates the time consuming stage of computing the *Self Forces*.

## 5 EXPERIMENTAL RESULTS

In this section three test scheduling examples are presented. The first one is a small example meant to give an idea about the type of results generated by PTS algorithms. Then a second example is discussed in order to provide a deeper insight into the results of PTS algorithms. Thus, a comparison of the power-test characteristics exhibited by their power-test scheduling solutions is discussed for a bigger block-test set example chosen randomly. In the end the PTS algorithms are compared in terms of "complexity vs results" for a block-test set example taken from [7]. This example is developed on the ASIC Z design discussed earlier on in [1] and [6]. Because idle blocks contribute very little to the total power dissipation, they are also excluded in this paper in order to simplify the calculations.

Suppose for the first example the following block-test set taken from figure 1. 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.

$t_1$ ( 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 scheduling chart solutions of the PTS-LS (PTS-LEA) and PTS-FDS (PTS-MSE) algorithms applied on this example without power constraints are depicted in figure 4. It can be seen in figure 4(b) that the PTS-FDS (PTS-MSE) approaches give a power-test chart solution exhibiting a more balanced power dissipation distribution than the PTS-LS (PTS-LEA) solution given in figure 4(a). Figure 5(a) depicts the power-test scheduling chart

| power constraints | PTS-LS scheduling | | | | | PTS-MSE scheduling | | | | | PTS-FDS scheduling | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TL | MPD | AVPD | PDD | RMS | TL | MPD | AVPD | PDD | RMS | TL | MPD | AVPD | PDD | RMS |
| 200 | 103 | 197 | 62.26 | 134.74 | 88.86 | 95 | 121 | 67.51 | 53.49 | 78.46 | 95 | 116 | 67.51 | 48.49 | 74.57 |
| 190 | 103 | 189 | 62.26 | 126.74 | 87.47 | 95 | 121 | 67.51 | 53.49 | 78.46 | 95 | 116 | 67.51 | 48.49 | 74.57 |
| 180 | 99 | 180 | 64.78 | 115.22 | 87.5 | 95 | 121 | 67.51 | 53.49 | 78.46 | 95 | 116 | 67.51 | 48.49 | 74.57 |
| 170 | 99 | 169 | 64.78 | 104.22 | 85 | 95 | 121 | 67.51 | 53.49 | 78.46 | 95 | 116 | 67.51 | 48.49 | 74.57 |
| 160 | 99 | 160 | 64.78 | 95.22 | 83.39 | 95 | 121 | 67.51 | 53.49 | 78.46 | 95 | 116 | 67.51 | 48.49 | 74.57 |
| 150 | 99 | 150 | 64.78 | 85.22 | 81.91 | 95 | 121 | 67.51 | 53.49 | 78.46 | 95 | 116 | 67.51 | 48.49 | 74.57 |
| 140 | 103 | 139 | 62.26 | 76.74 | 78.72 | 95 | 121 | 67.51 | 53.49 | 78.46 | 95 | 116 | 67.51 | 48.49 | 74.57 |
| 130 | 99 | 130 | 64.78 | 65.22 | 77.89 | 88 | 130 | 72.88 | 57.13 | 83.93 | 95 | 116 | 67.51 | 48.49 | 74.57 |
| 120 | 100 | 120 | 64.13 | 55.87 | 76.58 | 94 | 118 | 68.22 | 49.78 | 78.75 | 95 | 118 | 67.51 | 50.49 | 75.48 |
| 110 | 107 | 110 | 59.93 | 50.07 | 72.18 | 103 | 110 | 62.26 | 47.74 | 72.7 | 94 | 110 | 68.22 | 41.78 | 76.78 |
| 100 | 108 | 100 | 59.38 | 40.62 | 69.15 | 103 | 99 | 62.26 | 36.74 | 70.06 | 100 | 100 | 64.13 | 35.87 | 71.25 |
| 90 | 107 | 90 | 59.93 | 30.07 | 67.67 | 113 | 90 | 56.75 | 33.25 | 63.74 | 107 | 90 | 59.93 | 30.07 | 66.01 |
| 80 | 109 | 80 | 58.83 | 21.17 | 64.33 | 110 | 79 | 58.3 | 20.7 | 63.21 | 109 | 80 | 58.83 | 21.17 | 63.55 |
| 70 | 125 | 70 | 51.3 | 18.7 | 56.36 | 123 | 70 | 52.14 | 17.86 | 56.42 | 117 | 70 | 54.81 | 15.19 | 58.22 |
| 60 | 139 | 60 | 46.14 | 13.86 | 49.66 | 135 | 60 | 47.5 | 12.5 | 50.76 | 128 | 60 | 50.1 | 9.9 | 51.79 |
| 50 | 151 | 50 | 42.47 | 7.53 | 44.42 | 155 | 50 | 41.37 | 8.63 | 43.78 | 156 | 50 | 41.11 | 8.89 | 43.29 |
| 40 | 183 | 40 | 35.04 | 4.96 | 36.4 | 181 | 40 | 35.43 | 4.57 | 36.47 | 182 | 40 | 35.24 | 4.76 | 36.18 |
| 30 | 234 | 30 | 27.41 | 2.59 | 27.81 | 235 | 30 | 27.29 | 2.71 | 27.77 | 235 | 30 | 27.29 | 2.71 | 27.75 |
| 20 | 345 | 20 | 18.59 | 1.41 | 18.83 | 340 | 20 | 18.86 | 1.14 | 19.05 | 340 | 20 | 18.86 | 1.14 | 18.83 |

**Table 1. Comparison of Power-Test Characteristics for the PTS Approaches (Second Example)**

| power constraints | PTS-LEA1(MRU) scheduling | | | | | PTS-LEA2(LRU) scheduling | | | | | PTS-LEA3(RAND) scheduling | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TL | MPD | AVPD | PDD | RMS | TL | MPD | AVPD | PDD | RMS | TL | MPD | AVPD | PDD | RMS |
| 200 | 99 | 197 | 64.78 | 132.22 | 90.88 | 84 | 172 | 76.35 | 95.65 | 93.31 | 103 | 197 | 62.26 | 134.74 | 88.86 |
| 190 | 99 | 189 | 64.78 | 124.22 | 89.66 | 84 | 172 | 76.35 | 95.65 | 93.31 | 99 | 189 | 64.78 | 124.22 | 89.49 |
| 180 | 99 | 180 | 64.78 | 115.22 | 87.77 | 84 | 172 | 76.35 | 95.65 | 93.31 | 99 | 180 | 64.78 | 115.22 | 87.77 |
| 170 | 99 | 170 | 64.78 | 105.22 | 85.51 | 86 | 170 | 74.57 | 95.43 | 92.01 | 99 | 169 | 64.78 | 104.22 | 85.18 |
| 160 | 99 | 160 | 64.78 | 95.22 | 84.12 | 93 | 158 | 68.96 | 89.04 | 85.09 | 99 | 160 | 64.78 | 95.22 | 83.99 |
| 150 | 99 | 150 | 64.78 | 85.22 | 82.29 | 89 | 148 | 72.06 | 75.94 | 85.44 | 99 | 150 | 64.78 | 85.22 | 82.11 |
| 140 | 99 | 139 | 64.78 | 74.22 | 80.74 | 89 | 140 | 72.06 | 67.94 | 83.79 | 99 | 139 | 64.78 | 74.22 | 80.43 |
| 130 | 99 | 130 | 64.78 | 65.22 | 78.3 | 97 | 130 | 66.11 | 63.89 | 79.56 | 99 | 130 | 64.78 | 65.22 | 78.3 |
| 120 | 100 | 120 | 64.13 | 55.87 | 76.87 | 97 | 119 | 66.11 | 52.89 | 78.92 | 100 | 120 | 64.13 | 55.87 | 76.64 |
| 110 | 107 | 110 | 59.93 | 50.07 | 72.62 | 106 | 110 | 60.5 | 49.5 | 71.2 | 107 | 110 | 59.93 | 50.07 | 72.6 |
| 100 | 108 | 100 | 59.38 | 40.62 | 69.59 | 112 | 100 | 57.26 | 42.74 | 68.06 | 108 | 100 | 69.38 | 40.62 | 69.19 |
| 90 | 107 | 90 | 59.93 | 30.07 | 67.75 | 120 | 89 | 53.44 | 35.56 | 62.14 | 107 | 89 | 59.93 | 30.07 | 67.75 |
| 80 | 109 | 80 | 58.83 | 21.17 | 64.56 | 129 | 80 | 49.71 | 30.29 | 57.26 | 109 | 80 | 58.83 | 21.17 | 64.56 |
| 70 | 125 | 70 | 51.3 | 18.7 | 56.36 | 123 | 70 | 52.14 | 17.86 | 56.66 | 125 | 70 | 51.3 | 18.7 | 56.36 |
| 60 | 139 | 60 | 46.14 | 13.86 | 49.66 | 139 | 60 | 46.14 | 13.86 | 59.66 | 139 | 60 | 46.14 | 13.86 | 49.66 |
| 50 | 151 | 50 | 42.47 | 7.53 | 44.42 | 152 | 50 | 42.19 | 7.81 | 44.19 | 151 | 50 | 42.47 | 7.53 | 44.42 |
| 40 | 183 | 40 | 35.04 | 4.96 | 36.4 | 184 | 40 | 34.85 | 5.15 | 36 | 183 | 40 | 35.04 | 4.96 | 36.4 |
| 30 | 234 | 30 | 27.41 | 2.59 | 27.81 | 234 | 30 | 27.41 | 2.59 | 27.81 | 234 | 30 | 27.41 | 2.59 | 27.81 |
| 20 | 345 | 20 | 18.59 | 1.41 | 18.83 | 345 | 20 | 18.59 | 1.41 | 18.83 | 345 | 20 | 18.59 | 1.41 | 18.83 |

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

solution generated by the PTS-LS algorithm for a power dissipation constraint $P_{MAX} = 12$. Figure 5(b) gives the power-test characteristics for the same power-test scheduling chart. The following abbreviations have been used: 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 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. It is calculated as the ratio between the power-test area, taken up by the chart (see figure 5(b)), and TL. The rectangle given by AVPD and TL would be the ideal power-test scheduling chart and, therefore, the ideal test schedule 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 pover-test scheduling chart.

Further on, the PTS algorithms have been 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 approaches from a bigger solution space.

| power constraints | PTS-LS scheduling | | | | | PTS-MSE scheduling | | | | | PTS-FDS scheduling | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TL | MPD | AVPD | PDD | RMS | TL | MPD | AVPD | PDD | RMS | TL | MPD | AVPD | PDD | RMS |
| 900 | 300 | 605 | 335.42 | 269.58 | 386.2 | 320 | 465 | 314.46 | 150.54 | 348.81 | 300 | 464 | 335.42 | 128.58 | 366.21 |
| 800 | 300 | 605 | 335.42 | 269.58 | 386.2 | 320 | 465 | 314.46 | 150.54 | 348.81 | 300 | 464 | 335.42 | 128.58 | 366.21 |
| 700 | 300 | 605 | 335.42 | 269.58 | 386.2 | 320 | 465 | 314.46 | 150.54 | 348.81 | 300 | 464 | 335.42 | 128.58 | 366.21 |
| 600 | 300 | 584 | 335.42 | 248.58 | 372.15 | 320 | 464 | 314.16 | 149.54 | 352.19 | 300 | 464 | 335.42 | 128.58 | 366.21 |
| 500 | 300 | 464 | 335.42 | 128.58 | 366.2 | 320 | 464 | 314.46 | 149.54 | 352.19 | 300 | 464 | 335.42 | 128.58 | 366.21 |
| 400 | 360 | 400 | 279.52 | 120.48 | 295.17 | 360 | 387 | 279.52 | 107.48 | 286.53 | 360 | 387 | 279.52 | 107.48 | 289.33 |
| 300 | 400 | 296 | 251.57 | 44.44 | 257.89 | 400 | 296 | 251.57 | 44.44 | 256.76 | 400 | 296 | 251.57 | 44.44 | 258.13 |
| 200 | 670 | 196 | 150.19 | 45.81 | 152.12 | 670 | 176 | 150.19 | 25.81 | 151.04 | 670 | 176 | 150.19 | 25.81 | 151.04 |

**Table 3. Comparison of Power-Test Scheduling Characteristics (Third Example)**

| power constraints | PTS-LEA1(MRU) scheduling | | | | | PTS-LEA2(LRU) scheduling | | | | | PTS-LEA3(RAND) scheduling | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TL | MPD | AVPD | PDD | RMS | TL | MPD | AVPD | PDD | RMS | TL | MPD | AVPD | PDD | RMS |
| 900 | 300 | 652 | 335.42 | 316.58 | 388.63 | 300 | 605 | 335.42 | 269.58 | 386.87 | 300 | 605 | 335.42 | 269.58 | 387.62 |
| 800 | 300 | 652 | 335.42 | 316.58 | 388.63 | 300 | 605 | 335.42 | 269.58 | 386.87 | 300 | 605 | 335.42 | 269.58 | 387.27 |
| 700 | 300 | 652 | 335.42 | 316.58 | 388.63 | 300 | 605 | 335.42 | 269.58 | 386.87 | 300 | 605 | 335.42 | 269.58 | 387.29 |
| 600 | 300 | 594 | 335.42 | 258.58 | 374.49 | 300 | 584 | 335.42 | 248.58 | 372.51 | 300 | 584 | 335.42 | 248.58 | 372.56 |
| 500 | 300 | 494 | 335.42 | 158.58 | 367.35 | 300 | 474 | 335.42 | 138.58 | 366.55 | 300 | 464 | 335.42 | 128.58 | 366.99 |
| 400 | 360 | 400 | 279.52 | 120.48 | 296.13 | 360 | 400 | 279.52 | 120.48 | 294.22 | 360 | 400 | 279.52 | 120.48 | 295.66 |
| 300 | 400 | 296 | 251.57 | 44.44 | 257.89 | 400 | 298 | 251.57 | 46.44 | 257.1 | 400 | 296 | 251.57 | 44.44 | 257.75 |
| 200 | 670 | 196 | 150.19 | 45.81 | 152.12 | 670 | 196 | 150.19 | 45.81 | 151.79 | 670 | 196 | 150.19 | 45.81 | 152.11 |

**Table 4. Comparison of Power-Test Scheduling Characteristics (Third Example)**

In figure 6 these characteristics are generated for a range of power dissipation constraints from totally relaxed to fully tight. In table 1 the same characteristics are given for PTS-LS, PTS-MSE and PTS-FDS. In table 2 are listed the characteristics of the first, second and third algorithms of the PTS-LEA approach using, respectively, most recently used (MRU), least recently used (LRU), and random (RAND) insertion approaches [9]. Comparing the PTS-LS (PTS-LEA) scheduling solutions on the one hand with the PTS-FDS (PTS-MSE) results on the other hand, one can see that the former gives "noisier" solutions for relaxed constraints. That is, the PTS-LS characteristics (TL, MPD, AVPD, PDD, RMS) in figure 6(a) do not have a smooth trend while the power constraints are ranged from relaxed to tight. Intuitively, this "noise" is due to the lack of a global optimization function of PTS-LS-like approaches. On the other hand, the global priority function helps the PTS-FDS (PTS-MSE) approaches to have a global view over the solution space and to pick up most of the times better solutions. Therefore, the characteristics of the solutions given in figure 6(b) for the PTS-FDS (PTS-MSE) approaches are smoother. Overall, the PTS-FDS (PTS-MSE) approaches give out power-test scheduling profiles which exhibit more balanced power dissipation distributions as was the case in the example from figure 4. That is, the solutions given by PTS-FDS (PTS-MSE) algorithms have smaller PDD and MPD characteristics, but they are computationally more expensive than the PTS-LS approach. Moreover, if the PTS-FDS approach is computationally slightly more expensive than the PTS-MSE approach, the solutions given by the former are slightly more balanced. At the same time, all PTS approaches exhibit almost the same scheduling solutions for tight power dissipation limits when the solution space is actually narrowed. It can also be seen in figure 6 that the power-constrained test schedules have globally the same trends in both approaches, when ranging the power dissipation constraint. TL grows almost linearly with the power dissipation constraint increase, whereas MPD and PDD exhibit a linear decrease.

For the third example a practical testbench is taken into consideration. An extended case [7] of the ASIC Z design given in [1] is experimented with the PTS approaches. The testbench has 27 tests spread over 9 cores. The results are given in tables 3 and 4 over a range of the power dissipation constraint. The same conclusions can be drawn as for 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, 6]. That is due to the fact that the test scheduling discipline assumed in [1, 6] is the *non-partitioned 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.

## 6 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. The power-constrained test scheduling problem considered here is modeled, by means of the tree growing technique, equivalent to the classical job scheduling problem under the constraint that some jobs must be executed exclusively (incompatible). Thus, the tree growing approach re-uses classical algorithms and provides

fast results. Classical HLS scheduling approaches (e.g, left-edge algorithm, list scheduling and force-directed scheduling) are proposed in this paper to the power-constrained test scheduling problem. Their polynomial complexity is beneficial to the system-level test scheduling problem. Even though they do not guarantee optimal block-test scheduling solutions, their 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.


### ACKNOWLEDGEMENTS

The authors would like to thank Ionel Todincă, Jeannette Janssen, and Reem Yassawi for their support with the mathematical aspects of this approach.


## 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] S. WANG AND S.K. GUPTA: **ATPG for Heat Dissipation Minimization During Test Application** - *IEEE Transactions on Computers*, Vol. 47, No. 2, pp. 256-262, Feb, 1998.

[4] V. DABHOLKAR, S. CHAKRAVARTY, I. POMERANZ, S. REDDY: **Techniques for Minimizing Power Dissipation in Scan and Combinational Circuits During Test Application** - *IEEE Transactions on Computers*, Vol. 17, No. 12, pp. 1325-1333, Dec, 1998.

[5] P. GIRARD, L. GUILLER, C. LANDRAULT, S. PRAVOSSOUDOVITCH: **A Test Vector Inhibiting Technique for Low Energy BIST Design** - *Proceedings of The VLSI Test Symposium*, pp. 407-412, 1999.

[6] 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.

[7] E. LARSSON, Z. PENG: **Test Infrastructure Design and Test Scheduling Optimization** - *Proceedings of The IEEE European Test Conference*, 2000.

[8] 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.

[9] V. MURESAN, X. WANG, V. MURESAN, M. VLADUTIU: **The Left Edge Algorithm and the Tree Growing Technique in Power-Constrainted Block-Test Scheduling** - *Proceedings of the 18th IEEE VLSI TEST SYMPOSIUM (VTS) 2000*, Montreal, Canada, May, 2000.

[10] V. MURESAN, X. WANG, V. MURESAN, M. VLADUTIU: **List Scheduling and Tree Growing Technique in Power-Constrained Block-Test Scheduling** - *Proceedings of the IEEE European Test Workshop (ETW) 2000*, Cascais, Portugal, May, 2000.

[11] P.G. PAULIN, J.P. KNIGHT : **Force-Directed Scheduling for the Behavioral Synthesis of ASICs** - *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 8, No. 6, pp. 661–679, Jun, 1989.

[12] V. MURESAN, X. WANG, V. MURESAN, M. VLADUTIU: **Distribution-Graph Based Approach and Tree Growing Technique in Power-Constrained Block-Test Scheduling** - submitted to the *IEEE Asian Test Symposium (ATS) 2000*, Taipei, Taiwan, Dec, 2000.


## A The PSEUDOCODE of the PTS-FDS ALGORITHM

**-sort all the block-tests by their mobility in two steps (test length, power consumption);**
**-initialize the $GrowingTree$, the $BlockTestList$ and the $GapsList$;**
**-while (there are unscheduled block-tests) do:**
**/*$BlockTestList$ is not empty*/ {**

- **evaluate time frames for all block-tests;**
- **while (there are block-tests having null time frames){**
  - **$CurTest$ = head of $BlockTestList$;**
  - **insert $CurTest$ as the tail of $GrowingTree$ roots (new test section);**
  - **make $CurTest$ "used";**
  - **remove $CurTest$ from $BlockTestList$;**
  - **generate a $TwinGap$ gap as the twin of $CurTest$;**
  - **insert $TwinGap$ into $GapsList$;**
  - **evaluate time frames for all block-tests;}/*while*/**
- **$CurTest$ = the head of $BlockTestList$;**
- **evaluate time frames for all block-tests;**
- **update power/concurrency distribution graphs;**
- **calculate $CurTest's$ Self Forces for every feasible test subsession assignment;**
- **add incompatibility forces to Self Forces;**
- **SCHEDULE $CurTest$ to the test subsession exhibiting the lowest Self Force at assignment;**

**-}/*while*/**