# List Scheduling and Tree Growing Technique in Power-Constrained Block-Test Scheduling

Valentin Mureşan, Xiaojun Wang

School of Electronic Engineering
Dublin City University
Dublin, Ireland

Valentina Mureşan, Mircea Vlăduţiu

Faculty of Computer Science
"Politehnica" University of Timişoara
Timişoara, România

**Abstract**

*The list scheduling approach is proposed in this paper to overcome the problem of unequal-length block-test scheduling under power dissipation constraints. An extended tree growing technique is also used in combination with the list scheduling algorithm in order to improve the test concurrency having assigned power dissipation limits. Test scheduling examples are discussed in order to make a comparison of this approach with previous approaches (left-edge algorithm).*

## 1 Introduction

VLSI devices running in test mode can consume 100 - 200% higher power than when running in normal mode [1]. The heat dissipated during test application is lately one of the major considerations in *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 employed together with the *list scheduling* algorithm, to search for power-constrained block-test schedule profiles in a polynomial time. The algorithm fully exploits test parallelism under power constraints. This is achieved by overlaying the block-test intervals of compatible subcircuits to test as many of them as possible concurrently as long as the maximum accumulated power dissipation does not go over the given limit. A *constant additive model* is employed for power estimation throughout the algorithm. Average, maximal and RMS power estimations can be employed.

## 2 Test Scheduling Problem

The components which are required to perform a test (test control logic, test pattern generators, signature analyzers, test buses) are known as *test resources* and they may be shared among the blocks under test (BUT). 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 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, if there are no resource conflicts between the steps. 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.

*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 $t_i$, then it is possible to determine whether they could run in parallel without any resource conflict. A pair of tests that can be run concurrently is said to be *compatible*. Each application of time compatible tests is called a *test session*, and the time required for a test session is referred to as *test length*. From this point of view, circuits fall, in general, into two classes: circuits in which all tests are equal in length, and circuits in which the tests are unequal in length.

## 3 Power-Constrained Test Scheduling

In practical circuits (e.g. MCMs) only a few blocks or modules are activated at a certain moment, under normal system operation, while other blocks are

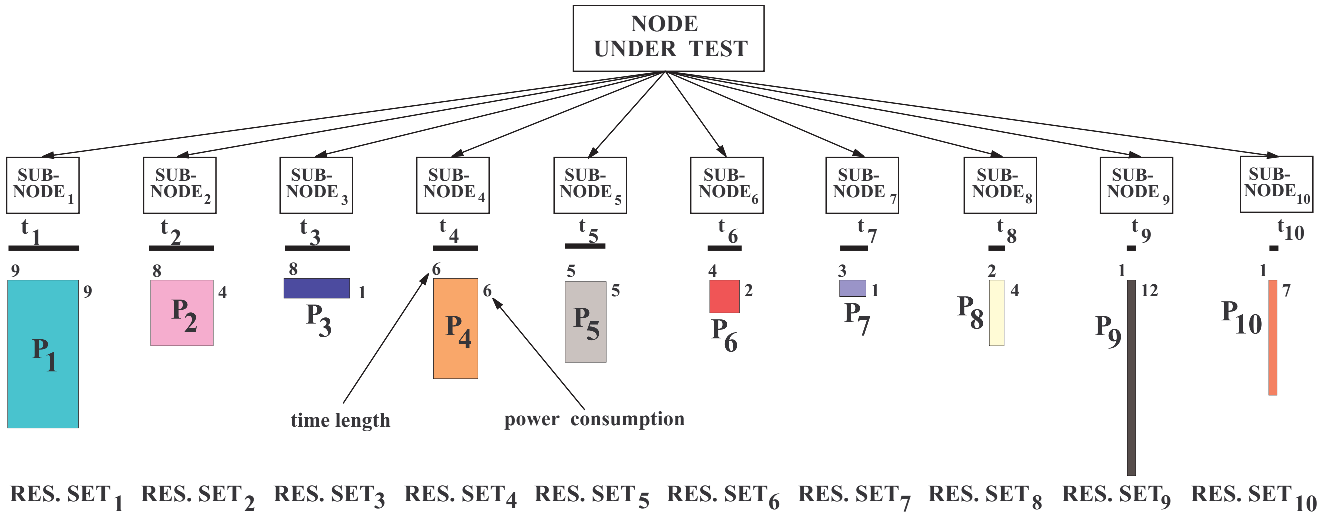


Figure 1: Example of Node Under Test

in the power-down mode to minimize the power dissipation. Under testing environment, however, in order to test the system in the shortest possible time, it is desirable to concurrently activate as many blocks as possible provided that the power dissipation limit of the system is not exceeded. 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 $t_j$. Usually this instantaneous power is constrained to not exceed the maximum 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. 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. Firstly, an overly optimistic and easy estimate $P(t_i)$ value can be defined as the *average power dissipation* over all test steps in $t_i$. Secondly, $P(t_i)$ can be defined as the *maximum power dissipation* over all test steps in $t_i$. This is the upper bound power dissipation in $t_i$ and its definition is pessimistic in this case since it disallows two tests $t_i$ and $t_j$, whose peak powers occur at different time instants, from being scheduled in the same test session. Thirdly, an *RMS power dissipation* can be employed when the instantaneous power dissipation is prone to power spikes and a more accurate estimation is sought.

Power dissipation during test scheduling was seldom under research so far. [2] proposes for the first time only a theoretical analysis of this problem at IC level. 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. Unfortunately, from an implementation point of view the identification of all cliques in the graph of compatible block-tests belongs to the class of NP-hard problems. Instead, a greedy approach is proposed in this paper. It has a polynomial complexity, which is very important for the success of a fast system-level test scheduling solution. A *list scheduling* together with a *tree growing technique* are employed here to generate the block-test schedule profile at the node level, within a test hierarchy.

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 node in hierarchy (apart from leaves) has different subnodes as children. Every test node $t_i$ 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 node. These parameters are the following: 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 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 dissipation constraints.

## 4 Tree Growing Technique

Since the block-test set in a complex VLSI circuit is huge, it is possible to schedule some short tests to begin, if they are resource compatible, when subcircuits with shorter testing time have finished testing, while other subcircuits with longer testing time have not. 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.

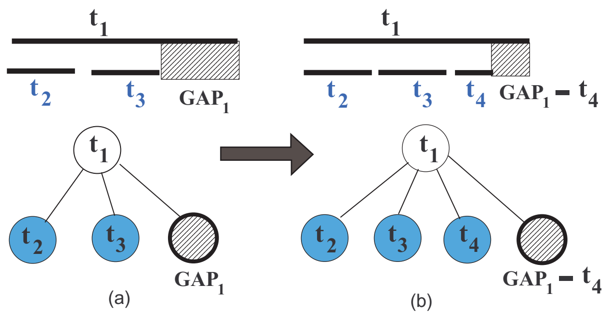


Figure 2: Merging Step Example

Nevertheless, 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 2 and 3. Therefore an *extended compatibility tree* (ECT), given by means of a *generalized tree*, is proposed here to overcome this problem. Figure 3 gives the *test schedule chart* and the ECT for the *power-test scheduling chart* example depicted in figure 4(a) and taken from [4]. In figures 4 and 5 the results of the PTS-LS algorithm are given both with (figures 4(a), 5(a)) and without (figures 4(b), 5(b)) power dissipation constraints ($P_{max} = 15$). The sequence of nodes contained in the same tree path represents an expansion of the CTS. Given a partial schedule chart of a CTS, a test $t$ can be merged into 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, if power dissipation constraint ($P_{max}$) is given, the power dissipation accumulated on the above tree path is less than or equal to $P_{max}$.

A *merging step* example is given in figure 2. Partial test schedule charts are given at the top, while partially grown compatibility trees are given at the bottom. Suppose tests $t_2$, $t_3$ and $t_4$ 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 to 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).

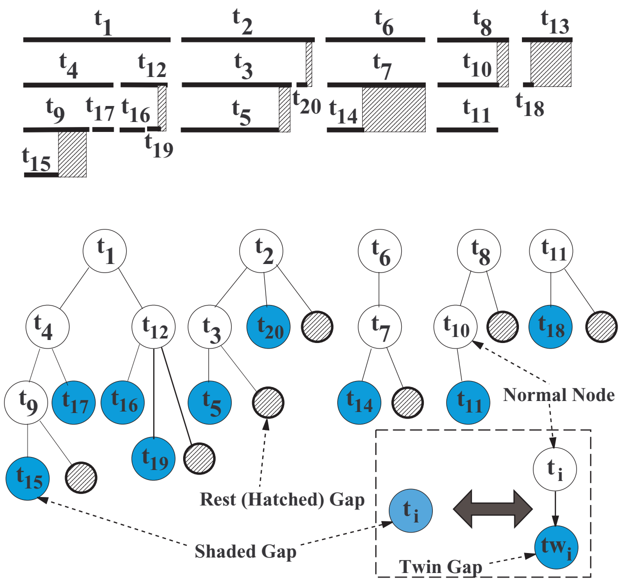
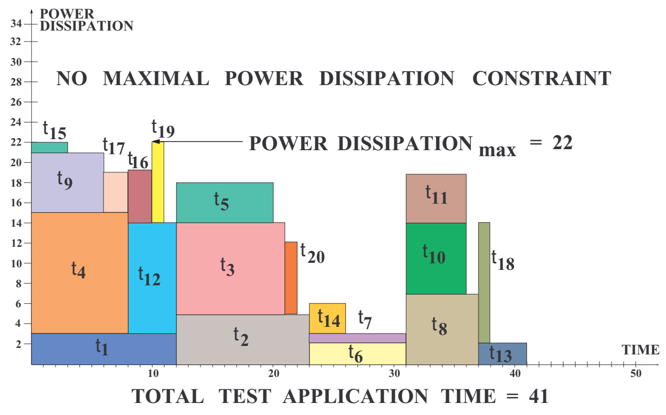
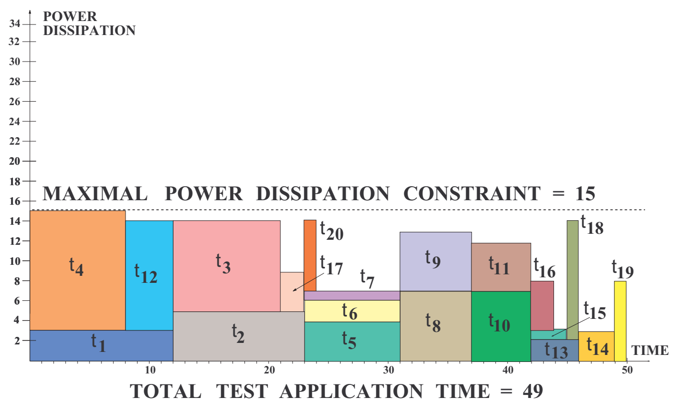


Figure 3: Tree Growing Example

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



(a) Without Power Dissipation Constraints



(b) With Maximal Power Dissipation Constraint = 15

Figure 4: Power-Test Scheduling Charts Of The List Scheduling Approach

leaf nodes within each ETP of ECT. These leaf nodes are called *gaps* and are depicted as either 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 cases of $GAP_1$ and $GAP_1 - t_4$ in figure 2. They are similar to the uncomplete branches of the binary tree in [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 bottom-right equivalence given in figure 3. The twin gaps 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. The root nodes (test sessions) are considered by default "shaded" gaps before any test subsession is generated below them. Finally, the test scheduling chart (figure 3) can easily be expanded to a power-test scheduling chart (figures 4) to asses the power dissipation distribution over the test schedule.

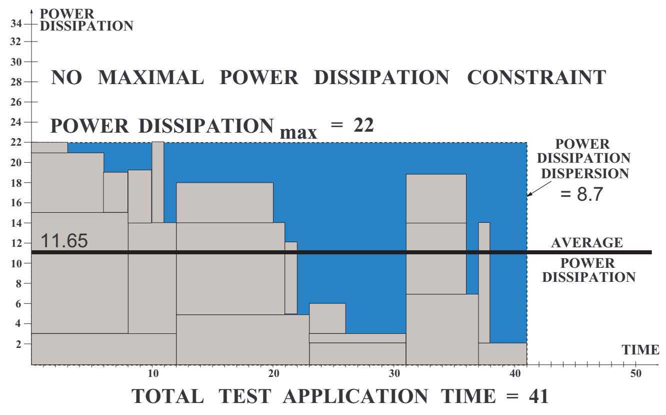
## 5 Power-Constrained Block-Test List Scheduling

The biggest achievement of the tree growing technique is that proven efficient HLS algorithms can be easily applied to the power-test scheduling (PTS) problem modeled as an extended tree growing process. This is due to the high degree of similarities between the HLS problems, e.g. HLS scheduling, and power-constrained block-test scheduling modeled as an growing tree problem. A classical HLS approach such as the left-edge algorithm for HLS register allocation has already been successfully applied on power-constrained block-test scheduling (PTS-LEA) in [5]. In this paper the efficiency of the HLS list scheduling algorithm (HLS-LS) in test scheduling is presented and is named *power-constrained block-test list scheduling* (PTS-LS).

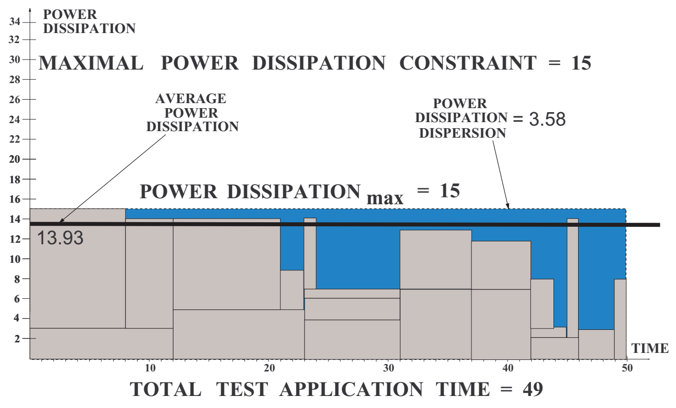
A clear parallel between the HLS scheduling problem and the PTS problem is given by the similarities between the control steps (c-step) in HLS and the test sessions (subsessions) in PTS, between operations (HLS) and block-tests (PTS), and between hardware resource constraints (HLS) and power dissipation 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). In the current PTS-LS algorithm the block-tests are initially ordered (as described above) before being scheduled. 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 for the other test (sub)sessions (ETP) left for further expansion. In terms of test scheduling, the list scheduling approach given here (PTS-LS) resembles a lot the first algorithm of the PTS-LEA approach given in [5].

In the HLS-LS approach, operations were sorted in topological order by using control and data dependencies. The set of operations that could be placed in a c-step were then evaluated. These operations were called *ready* operations. If the number of ready operations of a single type exceeded the number of hardware modules available to perform them, then one or more operations had to be deferred. In the HLS-LS algorithms, the selection of the deferred operations was determined by a local priority function such as *mobility* or *urgency*.

Since every block-test $t_i$ in the approach proposed in this paper has a test length $T_i$ and a power dissipation $P_i$, a local priority function called *test mo-*



(a) Without Power Dissipation Constraints



(b) With Maximal Power Dissipation Constraint = 15

Figure 5: Power-Test Scheduling Charts' Characteristics

*bility* $TM_i$ can also be defined here for $t_i$ as the inverse of the product between its test length $T_i$ and its power dissipation $P_i$: $TM_i = \frac{1}{T_i * P_i}$. The mobility of a block-test $t_i$ is inversely proportional with its dimensions. In figure 1 the dimension of a block-test $t_i$ is the area of the rectangle having its test length $T_i$ and its power dissipation $P_i$ as sides. That is, the bigger the area of the rectangles associated in figure 1 with the block-tests, the smaller the mobility they get. Intuitively, the probability of scheduling a block-test $t_i$ into a test subsession is higher the higher the test mobility $TM_i$ is. Though, in the tree growing approach the block-tests in a ETP are monotonously growing in terms of test length. A block-test cannot be scheduled into an ETP, where the leaf's test length is shorter than the test length of the block-test to be scheduled. This is due to the fact that the block-tests have to be scheduled in an ETP in the order of their test lengths. Therefore the mobility in the PTS-LS approach is split into two, its test length component and its power dissipation component. Thus, in PTS-LS, 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.

## 6 Experimental Results

In order to provide a deeper insight into the results of this algorithm a comparison of the characteristics exhibited by the power-test schedules generated by the PTS-LS approach and the PTS-LEA approach [5] is given in this section. Figure 4 depicts the characteristics of the power-test scheduling charts given by the PTS-LS approach for the example given in [4]. The following abbreviations will be used from now on for these characteristics: test length (TL), maximal accumulated power dissipation (MPD), average power dissipation (APD) and power dissipation dispersion (PDD). TL represents the total test application time of the test scheduling solutions. MPD is the maximal power dissipation over the whole test length of the power-test schedule. APD is considered the ideal MPD when all the ETP's 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) of the ECT, and TL. The rectangle given by APD and TL would be the ideal power-test scheduling chart and, thus, the ideal test schedule profile. PDD is directly proportional to the accumulated power dissipation dispersion over the power-test scheduling chart, which is considered here 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 APD.

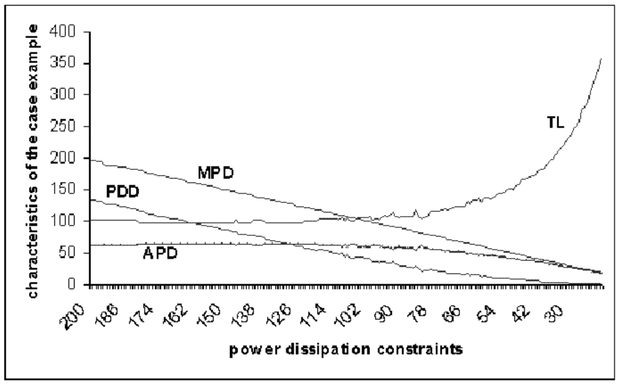


Figure 6: PTS-LS Solution Characteristics

It can be seen in figures 4(b), 5(b) and in table 1 that a tighter power dissipation constraint forces the

| COMPARISON | PTS-LS (ORDERED) | | | | PTS-LEA (MRU) | | | | PTS-LEA (RANDOM) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TL | MPD | APD | PDD | TL | MPD | APD | PDD | TL | MPD | APD | PDD |
| 200 | 103 | 197 | 62.26 | 134.74 | 111 | 113 | 57.77 | 55.23 | 100 | 120 | 64.13 | 55.87 |
| 180 | 99 | 180 | 64.78 | 115.22 | 111 | 113 | 57.77 | 55.23 | 85 | 156 | 75.45 | 80.55 |
| 160 | 99 | 160 | 64.78 | 95.22 | 111 | 113 | 57.77 | 55.23 | 99 | 159 | 64.78 | 94.22 |
| 140 | 103 | 139 | 62.26 | 76.74 | 111 | 113 | 57.77 | 55.23 | 91 | 139 | 70.47 | 68.53 |
| 120 | 100 | 120 | 64.13 | 55.87 | 111 | 113 | 57.77 | 55.23 | 108 | 120 | 59.38 | 60.62 |
| 100 | 108 | 100 | 59.38 | 40.62 | 125 | 95 | 51.3 | 43.7 | 111 | 100 | 57.77 | 42.23 |
| 80 | 109 | 80 | 58.83 | 21.17 | 125 | 79 | 51.3 | 27.7 | 122 | 80 | 52.57 | 27.43 |
| 60 | 139 | 60 | 46.14 | 13.86 | 127 | 60 | 50.5 | 9.5 | 132 | 60 | 48.58 | 11.42 |
| 40 | 183 | 40 | 35.04 | 4.96 | 189 | 40 | 33.93 | 6.07 | 183 | 40 | 35.04 | 4.96 |
| 20 | 345 | 20 | 17.91 | 2.09 | 356 | 20 | 18.01 | 1.99 | 346 | 20 | 18.53 | 1.47 |

Table 1: Comparison Of The Power-Test Scheduling Characteristics Given By The PTS Approaches

test scheduling to a more balanced power dissipation throughout the test application time. At the same time obvious power dissipation spikes could be seen in figures 4(a), 5(a) due to the lack of power dissipation constraints. That means the power dissipation is less balanced when it is loosely constrained. On the other hand when there are tighter power dissipation constraints (see table 1), the total test application time increases. Thus, this power-test scheduling problem turned out to be a trade-off problem to be solved with more complex algorithms for near-optimal solutions.

The PTS-LS and PTS-LEA [5] algorithms mentioned above 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. The case example of high resource compatibility degree has been chosen to run experiments in order to see the characteristics of the solutions chosen by the PTS-LS and PTS-LEA approaches from a big solution space. In figure 6 the curves of the TL, MPD, AVPD, PDD characteristics of the power-test scheduling charts given by the PTS-LS approach are depicted while the power dissipation constraints are ranged from relaxed to tight. It can be noticed that the PTS-LS scheduling approach gives "noisier" solutions for relaxed constraints. That is, the PTS-LS characteristics do not have a smooth trend while the constraints are ranged from relaxed to tight. Intuitively, the "noise" which appears amongst the characteristics of the solutions given by the PTS-LS approach is due to approach's lack of a global optimization function. A global optimization function would find most of the times more balanced solutions while the total test application time is kept tight as well. Table 1 gives a means of comparison of the characteristics of the power-test scheduling solutions found respectively by the PTS-LS, PTS-LEA (MRU) and PTS-LEA (RANDOM) approaches [5] for the same test scheduling example [4].

## 7 Conclusions

This novel greedy block-test scheduling approach is based on the classical list scheduling algorithm applied to an extended tree growing technique and its polynomial complexity is beneficial to the system-level test scheduling problem. Even though it does not guarantee optimal block-test scheduling solutions, its final result can be used as a starting point by near-optimal block-test scheduling approaches (i.e., simulated annealing, genetic algorithms, 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] 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.

[3] 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.

[4] V. MURESAN, X. WANG, V. MURESAN, M. VLADUTIU: **Power-Constrained Block-Test List Scheduling** - *Proceedings of the 11th IEEE International Workshop on Rapid System Prototyping*, accepted paper, Paris, June, 2000.

[5] 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*, accepted paper, Montreal, May, 2000.