Test Scheduling for Minimal Energy Consumption under Power Constraints

Tobias Schuele and Albrecht P. Stroele
University of Karlsruhe
Institute of Computer Design and Fault Tolerance (Prof. D. Schmid)
P. O. Box 6980, 76128 Karlsruhe, Germany
E-mail: schuele@ira.uka.de

Abstract

Power consumption has become a crucial concern in built-in self-test (BIST) due to the increased switching activity in the circuit under test. In this paper we present a method for scheduling tests which aims at minimizing total energy consumption and test application time under peak power constraints. In contrast to previous approaches, our method takes into account switching activity which occurs in overlapping regions of the subcircuits under test. The key part is a hierarchical approach to power estimation which makes it possible to quickly evaluate the power consumption of partial schedules. Experimental results show that the energy savings range between 54% and 97% in comparison with conventional methods. Test application time can be reduced to the same extent.

1. Introduction

In recent years, energy consumption and power dissipation have attained considerable attention in circuit design. Several factors have contributed to this trend. With the advent of portable devices, for example, low energy consumption has become one of the major design goals in order to prolong battery life. Moreover, the amount of energy a circuit consumes is directly reflected in its heat dissipation. Excessive heat dissipation, however, requires expensive packaging and cooling techniques which in turn increase system cost. In addition, as power consumption increases, circuit reliability gets affected adversely due to electro-migration.

At the same time, the complexity of state-of-the-art circuits limits their testability by means of external test equipment. Consequently, built-in self-test (BIST) has emerged as a favorable method for circuit testing. The integration of test functionality on the chip has several advantages such as improved testability, no need for external test hardware, and the ability to test a circuit in the field. Also, BIST allows to preserve the intellectual property in core-based designs.

In test-per-clock BIST schemes, a test pattern can be applied to the circuit under test in each clock cycle. This enables the detection of delay-dependent faults and reduces test application time. Usually, not all subcircuits can be tested in parallel due to resource conflicts. Hence, the tests must be scheduled in a way such that no two conflicting tests are executed at the same time. In the following, we will concentrate on test-per-clock BIST schemes that can be implemented using BILBO-like test registers [9]. Methods for test register insertion at register transfer and at gate level have been described in [1, 10, 12, 13].

During normal mode of operation, the input vectors which are applied to a circuit are usually strongly correlated. When the circuit is tested with pseudo-random patterns, however, consecutive input vectors are statistically independent which results in increased switching activity in the circuit under test. Since in CMOS circuits energy is primarily consumed by signal transitions, the average power consumption during testing is significantly higher than in normal mode of operation [16].

Another important aspect is the peak power consumption during BIST. Usually, as many compatible tests as possible are performed concurrently in order to minimize the total test application time. As a result, much more subcircuits may be active simultaneously compared to normal mode of operation. The specified power rating may be exceeded and thus the circuit under test gets damaged.

Recently, test scheduling methods have been proposed that minimize the total test application time under power constraints [4, 11]. Given the test resources of a circuit, a test compatibility graph (TCG) can be constructed which models the dependencies between different tests. Two nodes in the TCG are connected by an edge if and only if the corresponding tests can be executed concurrently. Consequently, the cliques of the TCG represent the maximum sets of compatible tests. However, executing all the tests of a clique in parallel may exceed the power limit. For this reason, the authors of [4] propose an improved scheduling strategy: In the first step, a set of power compatible subsets is extracted from each clique. A power compatible set (PCS) is a set of tests which can be executed concurrently and which comply with the imposed power limit. Next, a minimum cover of the set of PCSs is computed such that every test is performed at least once.

The heuristic algorithm of [11] has polynomial time complexity. Dealing with tests of unequal lengths, it builds test schedules where the processing of the tests may finish...
at many different points of time. This can further reduce the energy consumption and the total test application time, but test control becomes more complex than with [4].

Both methods are based on the concept of block-tests as proposed in [7]. They assume that the subcircuits under test are separated blocks which do not influence one another so that the power consumption of blocks tested concurrently can simply be summed up. This assumption simplifies test scheduling but can lead to a waste of energy. Consider the example in figure 1. There are two overlapping subcircuits $c_1$ and $c_2$ with the response compactors $t_1$ and $t_2$, respectively. The shaded area is included in both subcircuits. The test registers $t_3, \ldots, t_5$ serve as pattern generators. Both subcircuits share the pattern generator $t_6$. Hence, switching activity occurs also in $c_2$ when $c_1$ is tested and vice versa.

![Figure 1: Overlapping subcircuits ($c_1$, $c_2$) with response compactors ($t_1$, $t_2$) and pattern generators ($t_3, \ldots, t_5$)](image)

More in detail, when subcircuit $c_1$ is tested, the gates and flip-flops in the shaded area are exercised. The same holds when subcircuit $c_2$ is tested. If both subcircuits are tested simultaneously, the energy for testing the shared, shaded part is required only once. However, if the subcircuits are tested one after the other, the energy consumed in the shaded area is required twice.

Furthermore, when a subcircuit is tested, adjacent logic blocks may also get some switching activity. For the subcircuit with response compactor $t_1$, this is indicated by the arrows in figure 1. This switching activity does not contribute to an improved fault coverage and thus energy is wasted. However, if the two neighboring subcircuits $c_1$ and $c_2$ are tested in parallel, part of this energy can be saved.

Overlaps between the subcircuits under test are also important when determining the peak power consumption for a set of tests. Suppose that $P_{\text{peak}}(\{c_1\}) = 10$ and $P_{\text{peak}}(\{c_2\}) = 12$ for the example in figure 1. Moreover, let the power limit be 20. If we do not account for the intersection between $c_1$ and $c_2$, i.e., if we assume that $P_{\text{peak}}(\{c_1, c_2\}) = P_{\text{peak}}(\{c_1\}) + P_{\text{peak}}(\{c_2\})$, then $c_1$ and $c_2$ should not be tested simultaneously. This is an overly pessimistic power estimate, however, since the intersection between the two subcircuits is considered twice. If the peak power consumption in the shaded area is at least 2, both subcircuits can be tested simultaneously without exceeding the power limit.

As the block-test scheduling algorithms of [4] and [11] do not exploit any information about overlaps and interactions between the subcircuits, they cannot prevent that energy is consumed multiple times in one and the same region. Even if two overlapping or interacting subcircuits are selected by chance for the same test session, this may un-true be rejected because of a power estimate exceeding the power limit.

In this paper we present a method for power efficient test scheduling which copes with the above mentioned problems. Its objective is to minimize the overall energy consumption and to ensure that power peaks which occur during testing do not violate the circuit specification. The rest of this paper is organized as follows: In the next section, we formulate the problem of finding power efficient test schedules in the context of test-per-clock BIST schemes. In section 3 we introduce the notion of activity regions which partition the circuit and thereby simplify power estimation (section 4). The scheduling algorithm and experimental results are then presented in sections 5 and 6. Finally, section 7 summarizes the presented concepts.

2. Problem Formulation

In test-per-clock BIST schemes, the circuit under test is segmented into a set of subcircuits which are completely bounded by test registers (figure 2). For that purpose, some conventional registers are enhanced to test registers which generate patterns and compact test responses in test mode. Additionally, test registers are added at the primary inputs and outputs of the circuit. In order to test one of these subcircuits, at least one test register must collect test responses (Multiple Input Signature Register, MISR). Thus, the smallest unit that can be tested stand-alone comprises one test register which can be configured as a MISR, the logic blocks which are connected to the inputs of this register, and a set of test registers which act as pattern generators. In this way, every test unit $u_2$ is uniquely determined by a single test register $t_1$. As figure 2, for example, test unit $u_2$ consists of the MISR $t_2$, the logic blocks $b_2$ and $b_3$, and the pattern generators $t_6$ and $t_7$.

![Figure 2: Circuit with logic blocks ($b_1, \ldots, b_5$), test registers ($t_1, \ldots, t_6$), and sample test unit ($u_2$)](image)

Usually, test registers like BILBOs cannot generate patterns and compact responses at the same time. Consequently, if one test unit employs a test register as a pattern generator and another test unit uses the same register as a
3. Circuit Partitioning

Due to the fact that test units may overlap, it does not suffice to estimate the power consumption for each test unit separately without considering overlaps and interactions between the subcircuits under test. On the other hand, it is much too complex and time consuming to repeat the complete power estimation procedure for all possible combinations of test units during scheduling. In order to get the required power estimates in an efficient way, we developed a hierarchical approach. We partition the circuit into activity regions that have a finer granularity than the test units but often contain a large number of gates and flip-flops. Then we estimate the power consumption for each activity region in the context of all the test units that can cause switching activity in the considered region. Based on these values, power estimates for all test units and arbitrary test sessions can be computed quickly.

The activity regions $A = \{a_1, \ldots, a_n\}$ of a circuit with built-in test registers are defined as the maximal subcircuits with the following properties:

(i) For each test unit $u_i \in U$ and each activity region $a_r \in A$: All the gates and flip-flops in $a_r$ are included in $u_i$, or none of them.

(ii) For each test unit $u_i \in U$ and each activity region $a_r \in A$: When $u_i$ is tested, all the gates and flip-flops in $a_r$ are reachable from the pattern generators of $u_i$ and thus can have some switching activity, or none of them.

(iii) Every test register is an activity region of its own.

The activity regions form a partition of the circuit, and every test unit can be described as a union of connected activity regions. For each test unit, the fan-out region of the pattern generators includes all the parts of the circuit that can have some switching activity when this unit is tested. Intersecting the fan-out regions of different test units gives the partitioning according to (ii). The partitioning according to (i) can be found by intersecting test units.

The algorithm in figure 3 outlines a procedure to determine the activity regions of a given circuit. It operates on a graph description of the circuit at gate level and has polynomial time complexity. For each test unit, the algorithm traverses the graph in backward and forward direction and marks all visited nodes with two different kinds of labels. The labels assigned during backward traversal indicate to which test units a node belongs to. The labels assigned during forward traversal show the test units where the marked nodes can have some switching activity. Nodes with the same set of labels form an activity region.

```
/* Input: Set of test units, U */
/* Output: Set of activity regions, A */

for each test unit $u_i \in U$
    traverse the circuit graph in backward direction from the inputs of the pattern generators to the outputs of other test registers and mark all nodes with label $b_i$

for each test unit $u_i \in U$
    traverse the circuit graph in forward direction from the outputs of the pattern generators to the inputs of other test registers and mark all nodes with label $f_i$

Collect all the nodes that have the same set of b-labels and f-labels and thus belong to the same activity region.

for each test register
    form a separate activity region
```

Figure 3: Algorithm for computing activity regions

In figure 4, the circuit whose block diagram is shown in figure 2 is partitioned into activity regions. The logic block $b_1$ is split up into two separate activity regions $a_1$ and $a_2$. This is because switching activity occurs in $a_2$ but not in $a_1$ when test unit $u_2$ or $u_3$ is tested. Conversely, assuming that
the test register \( r_j \) influences the entire block \( b_4 \), the logic blocks \( b_4 \) and \( b_5 \) form a single activity region \( a_5 \). In this case switching activity occurs always in both \( b_4 \) and \( b_5 \).

![Figure 4: Circuit of figure 2 partitioned into activity regions \((a_1, \ldots , a_6)\)](image)

### 4. Power Estimation

After partitioning the circuit, we estimate the power for each activity region. For that purpose, we use a dynamic power estimation technique based on Monte Carlo methods as described in [3]. A power estimate is obtained by simulating the circuit with a set of typical input patterns and counting the number of signal transitions. Note that in CMOS circuits, static sources of power consumption such as leakage currents can be neglected.

Let \( n_i(N) \) be the number of transitions on net \( i \) during the clock cycles \( 0, \ldots , N \) and \( \alpha_i = \lim_{N \to \infty} \frac{n_i(N)}{N} \) the average number of transitions per clock cycle. Then the average power which is consumed by a circuit with \( n \) nets can be computed as follows:

\[
P = \frac{1}{2} V_{dd}^2 f_{clk} \sum_{i=1}^{n} \alpha_i C_i
\]

where \( V_{dd} \) is the supply voltage, \( f_{clk} \) the clock frequency, and \( C_i \) the capacitance of net \( i \). In the following we will restrict ourselves to the weighted switching activity (WSA) as a measure of the energy consumption. It is defined as the sum of \( n_i(N)C_i \) over all circuit nodes where \( N \) represents the total number of clock cycles during simulation. We assume that the capacity of a net is proportional to the number of gates it connects. Similarly, we use a real delay model where the delay of a gate is proportional to its fanout and short pulses are filtered out (inertial delay).

For each test unit we perform a separate simulation run and measure the power consumption of all affected activity regions. Let \( P_{avg}(a_r)|_{u_i} \) denote the average power consumption of activity region \( a_r \) when test unit \( u_i \) is tested. Then the average power of testing just unit \( u_i \) can easily be determined by summing over all activity regions:

\[
P_{avg}(u_i) = \sum_r P_{avg}(a_r)|_{u_i}
\]

To calculate the average power of a test session \( s_j \), no further simulation runs are required. It suffices to sum up the contributions of all activity regions which are influenced by the pattern generators used in \( s_j \). There are three cases we must distinguish:

- If an activity region \( a_r \) is included in a test unit \( u_i \in s_j \), then we take the value \( P_{avg}(a_r)|_{u_i} \). Similarly, if \( a_r \) is not part of any test unit of \( s_j \), but all pattern generators which influence \( a_r \) are active during session \( s_j \), we take \( P_{avg}(a_r)|_{u_h} \) for a test unit \( u_h \notin s_j \) which includes \( a_r \).
- If \( a_r \) is influenced (but not included) by only one test unit \( u_i \) or by several test units in the same way, then we take \( P_{avg}(a_r)|_{u_i} \).
- If in the rare case where several test units influence \( a_r \) in different ways, but still not all the pattern generators which can affect \( a_r \) are active during session \( s_j \), we set the power estimate as obtained during simulation in relation to the number of pattern generators which actually cause switching activity in \( a_r \).

For the example of figure 4, the average power of the test session \( s_j = \{u_2, u_3\} \) is \( P_{avg}(a_2)|_{u_2} + P_{avg}(a_3)|_{u_2} + P_{avg}(a_4)|_{u_3} + P_{avg}(a_5)|_{u_3} \) plus the average power consumed in the test registers.

Peak power consumption can be computed in a similar way. First, the value \( P_{peak}(a_r)|_{u_i} \) is estimated for all test units and all affected activity regions. Then, the peak power of a test session \( s_j \) can be estimated by

\[
P_{peak}(s_j) = \sum_r \max_{u_i \in s_j} \{P_{peak}(a_r)|_{u_i}\}
\]

where the sum is taken over all the activity regions that have some switching activity in session \( s_j \).

The test application time of test session \( s_j \) is

\[
t(s_j) = \max_{u_i \in s_j} \{t(\{u_i\})\}
\]

Finally, the energy required by test session \( s_j \) is

\[
E(s_j) = P_{avg}(s_j) \cdot t(s_j).
\]

### 5. Test Scheduling

For many real-life circuits, the large number of test units makes an exhaustive exploration of all possible test schedules infeasible. More precisely, the test scheduling problem considered here (see section 2) is an NP-complete problem. For this reason, we developed an efficient greedy algorithm to construct near-optimal solutions where the test sessions are built one after another. In each step, a single test unit is added to the current session. The basic idea is to select that test unit which results in the largest energy savings.

Suppose that the overlapping test units \( u_3 \) and \( u_4 \) of figure 4 have already been included in the current test session \( s_j \). The inclusion of \( u_1 \) adds switching activity only in those
parts of the circuit that are not also exercised by \( u_2 \) and \( u_3 \), i.e., in activity region \( a_1 \) and partly in \( a_2 \). So the average power of \( s_j \) increases by \( P_{\text{avg}}(a_1)_{u_1} + P_{\text{avg}}(a_2)_{u_2} - P_{\text{avg}}(a_2)_{u_2} \) whereas a separate test session with \( u_3 \) would have average power \( \sum_{i=1}^{5} P_{\text{avg}}(a_i)_{u_i} \). As a result, energy can be saved by including \( u_1 \) in \( s_j \) provided that the test lengths of \( u_1 \) and \( s_j \) are roughly the same.

In general, the energy savings can be computed by means of the following formula:

\[
E_{\Delta}(u_i, s_j) = E(\{u_i\}) + E(s_j) - P_{\text{avg}}(\{u_i\} \cup s_j) \cdot \max\{t(\{u_i\}), t(s_j)\}
\]

On the other hand, the constraint on peak power may limit the number of test units that can be tested concurrently. Then, those test units should be preferred that cause only a small increase in peak power. To account for this aspect, the energy savings are divided by the increase in peak power.

The gain \( g(u_i, s_j) \) of adding a compatible test unit \( u_i \) to the (partial) test session \( s_j \) is thus determined by

\[
g(u_i, s_j) = \begin{cases} 
\frac{E_{\Delta}(u_i, s_j)}{P_{\text{peak}}(\{u_i\} \cup s_j) - P_{\text{peak}}(s_j)} & \text{if } P_{\text{peak}}(\{u_i\} \cup s_j) \leq P_{\text{lim}} \text{ and } u_i \text{ is compatible to all test units in } s_j \\
-\infty & \text{otherwise}
\end{cases}
\]

In each step, the test unit with maximal nonnegative gain is added to the current test session. If the power limit is exceeded or no more energy savings can be achieved in this way, the next test session is started. Initially, the test unit with the largest test length is included. The algorithm in figure 5 shows the complete scheduling procedure.

---

**Figure 5: Greedy algorithm for test scheduling**

---

### Table 1: Benchmark characteristics

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Pts</th>
<th>PEs</th>
<th>FFs</th>
<th>Gates</th>
<th>Test Units</th>
<th>Activity Regions</th>
<th>Test Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>s13207</td>
<td>31</td>
<td>121</td>
<td>669</td>
<td>7951</td>
<td>87</td>
<td>520</td>
<td>1599047</td>
</tr>
<tr>
<td>s15850</td>
<td>14</td>
<td>87</td>
<td>597</td>
<td>9772</td>
<td>93</td>
<td>906</td>
<td>2168232</td>
</tr>
<tr>
<td>s25932</td>
<td>35</td>
<td>320</td>
<td>1728</td>
<td>16085</td>
<td>65</td>
<td>314</td>
<td>2391019</td>
</tr>
<tr>
<td>s38417</td>
<td>28</td>
<td>106</td>
<td>1636</td>
<td>22177</td>
<td>160</td>
<td>1060</td>
<td>4440978</td>
</tr>
<tr>
<td>s38584</td>
<td>12</td>
<td>278</td>
<td>1452</td>
<td>19523</td>
<td>180</td>
<td>2089</td>
<td>5329035</td>
</tr>
<tr>
<td>b14s</td>
<td>33</td>
<td>54</td>
<td>245</td>
<td>4776</td>
<td>70</td>
<td>218</td>
<td>1115808</td>
</tr>
<tr>
<td>b15s</td>
<td>37</td>
<td>70</td>
<td>449</td>
<td>8944</td>
<td>83</td>
<td>355</td>
<td>1706255</td>
</tr>
<tr>
<td>b17s</td>
<td>38</td>
<td>97</td>
<td>1415</td>
<td>24116</td>
<td>209</td>
<td>1779</td>
<td>6033103</td>
</tr>
<tr>
<td>b13s</td>
<td>38</td>
<td>73</td>
<td>3230</td>
<td>66887</td>
<td>452</td>
<td>3635</td>
<td>13770349</td>
</tr>
<tr>
<td>b20s</td>
<td>33</td>
<td>22</td>
<td>490</td>
<td>9804</td>
<td>69</td>
<td>498</td>
<td>2294862</td>
</tr>
<tr>
<td>b21s</td>
<td>33</td>
<td>22</td>
<td>490</td>
<td>9804</td>
<td>69</td>
<td>493</td>
<td>2294848</td>
</tr>
<tr>
<td>b22s</td>
<td>33</td>
<td>22</td>
<td>735</td>
<td>15072</td>
<td>106</td>
<td>790</td>
<td>3409650</td>
</tr>
</tbody>
</table>

---

### 6. Experimental Results

The described procedures have been applied to the largest circuits of the ISCAS’89 [2] and the ITC’99 [6] benchmark sets. To implement BIST, we inserted test registers using a bottom-up approach which starts at the gate level. First, a number of flip-flops were selected such that all the cycles of the circuit structure are cut with minimum hardware overhead [13]. Then, the selected flip-flops were merged into test registers. Here, we tried to minimize the overlaps between the resulting test units. For each test unit, the test length was determined such that 100% fault coverage is achieved or at most 65536 patterns are applied.

Table 1 outlines the characteristics of the circuits including the number of test units and the number of activity regions. The last column shows the total test length of all test units, i.e., the test application time when all tests are performed one after another (sequential test schedule).
basis of activity regions are much larger and range between 57% (s13207) and 91% (b18s).

Figure 6: Relative energy consumption for schedules with minimal power limit

In a second set of experiments, we used our method to minimize the energy consumption with unconstrained peak power. The actual peak power values of the resulting schedules were then used as power limits for block–test scheduling. Thus, the constructed block–test schedules cannot have larger peak power than the schedules obtained by our method. The results are given in figure 7. The gain for the block–test scheduling approach is moderate in comparison to the case in figure 6. Again, for half of the circuits the energy reduction is negligible. With our method, the energy savings range from 88% to 97%.

Figure 7: Relative energy consumption for schedules with relaxed power limit

Moreover, we determined the test lengths subject to the scheduling strategy (table 2). For the schedules with minimal peak power, an average of 76% of the test length of the sequential schedule is required for the block–test approach as opposed to our method where the test lengths are reduced by more than a factor 5 on average. These results indicate that in the former case only few test units can be scheduled into the same session. In the case of relaxed power limit, the average test lengths amount to 63% and 4%, respectively.

Table 2: Relative test lengths for schedules with minimal and relaxed power limit

Furthermore, we are interested in the impact of different power limits on our scheduling strategy. For that purpose, we successively increased the power limit and observed the decrease in energy consumption and test length. Increasing the power limit allows more test units to be scheduled concurrently. Hence, correlations between different test units due to shared subcircuits can be exploited more efficiently. Figure 8 shows the results for circuit b18s. The starting point (100%) corresponds to the schedule constructed with minimal power limit. The figure demonstrates that increasing the power limit can reduce the energy consumption step by step. If we double the power limit, for example, the energy consumption approximately halves. The same holds for the test length even though it may increase temporarily. Note that the peak power gets saturated at about 240%. Higher values of the power limit cannot give further reductions in energy and test length.

Figure 8: Impact of power limit on energy consumption and test length of circuit b18s
Finally, we evaluated the accuracy of our power estimation procedure. For that purpose, we simulated the entire test sessions as obtained by the scheduling algorithm and compared the results with the estimated values (table 3). The estimates for the average power consumption are highly accurate. For most circuits, the deviation amounts to less than 2% (average over all test sessions). The estimates for the peak power consumption, however, exceed the actual values significantly. This is due to the worst case assumption that all the activity regions of a test session consume maximal power at the same time.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Average Power</th>
<th>Peak Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>s13207</td>
<td>-4.0%</td>
<td>1.6%</td>
</tr>
<tr>
<td>s15850</td>
<td>-14.7%</td>
<td>-0.6%</td>
</tr>
<tr>
<td>s35932</td>
<td>-5.4%</td>
<td>-1.2%</td>
</tr>
<tr>
<td>s38417</td>
<td>-6.9%</td>
<td>-1.1%</td>
</tr>
<tr>
<td>s38584</td>
<td>-1.9%</td>
<td>2.5%</td>
</tr>
<tr>
<td>b14s</td>
<td>-2.3%</td>
<td>-0.5%</td>
</tr>
<tr>
<td>b15s</td>
<td>-0.8%</td>
<td>2.1%</td>
</tr>
<tr>
<td>b17s</td>
<td>-5.1%</td>
<td>0.1%</td>
</tr>
<tr>
<td>b18s</td>
<td>-14.6%</td>
<td>-0.3%</td>
</tr>
<tr>
<td>b20s</td>
<td>-2.1%</td>
<td>-0.1%</td>
</tr>
<tr>
<td>b21s</td>
<td>-2.6%</td>
<td>1.0%</td>
</tr>
<tr>
<td>b22s</td>
<td>-2.4%</td>
<td>-0.6%</td>
</tr>
</tbody>
</table>

Table 3: Accuracy of power estimation on the basis of activity regions

7. Conclusions

In this paper we presented a method for scheduling tests under power constraints. It takes into consideration switching activity which occurs in overlapping regions of the subcircuits under test. As a result, tighter estimates on the power consumption are achieved which extend the scope for test scheduling. Moreover, energy is saved if test units which share common subcircuits or influence one another are executed in the same test session.

Experimental results show that the energy savings under minimal peak power constraints range between 54% and 89% for our method in comparison with a simple block-test scheduling approach. At the same time, the test length is reduced by 59% to 91%. If the power limit is relaxed, even larger reductions can be achieved for the total energy consumption (80%–97%) and the test length (77%–98%).

Our method can be combined with other approaches which aim at reducing the power consumption during built-in self-test such as pattern filtering [5], where only those patterns are passed to the circuit under test which actually contribute to an increased fault coverage. In [14], the number of transitions at the inputs of the circuit is reduced by operating part of the pattern generators in a slow mode. Similarly, it has been shown in [15] that weighted random patterns can be effective in reducing the energy consumption during BIST. Another strategy is to partition the circuit into independent subcircuits such that the switching activity per time interval is minimized [8].

References