Integrated Test Scheduling, Test Parallelization and TAM Design

Erik Larsson+, Klas Arvidsson*, Hideo Fujiwara* and Zebo Peng*

Computer Design and Test Laboratory+
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0101, Japan
{erila, fujiwara}@is.aist-nara.ac.jp

Embedded Systems Laboratory*
Linköpings Universitet
SE-582 83 Linköping, Sweden
{erila, zebpe}@ida.liu.se

Abstract

We propose a technique integrating test scheduling, scan chain partitioning and test access mechanism (TAM) design minimizing the test time and the TAM routing cost while considering test conflicts and power constraints. Main features of our technique are (1) the flexibility in modelling the systems test behaviour and (2) the support for interconnection test of unwrapped cores and user-defined logic. Experiments using our implementation on several benchmarks and industrial designs demonstrate that it produces high quality solution at low computational cost.

1 Introduction

The testing of System-on-Chip (SOC) designs is a crucial problem mainly due to the increasing design complexity, which leads to high test data volumes. High test data volumes are causing to two major problems depending on each other; (1) how design a minimal test access mechanism (TAM) and (2) how design a test schedule minimizing test time under test conflicts and power constraint.

The work-flow designing an SOC test solution can be divided into two consecutive parts; an early design space exploration followed by an extensive optimization for the final solution. For the former, we have proposed an integrated technique for test scheduling and TAM design minimizing test time and TAM cost [2]. The advantage of the technique is its low computational cost making it useful for iteratively use in the early design space exploration phase. For extensive optimization for the final solution, we have proposed a technique based on Simulated Annealing, which is used only a few times justifying its high computational cost. We have, independently, proposed an integrated test scheduling and scan chain partitioning (test parallelization) technique under power constraints [4].

In this paper, we integrate test scheduling, scan-chain partitioning and TAM design with the objective to minimize test time and TAM routing cost while considering test conflicts and power constraint. An aim is to reduce the gap between the design space exploration and the extensive optimization, i.e. produce a high quality solution in respect to test time and TAM cost at a low computational cost. We have analyzed our previous approach and noted the TAM design part is of low quality. Further features of our proposed approach are that we support:

- testing of user-defined logic (UDL),
- testing of unwrapped cores,
- memory limitations at test sources,
- bandwidth limitations on test sources and test sinks,
- embedding cores in core.

We have implemented our technique and performed several experiments on benchmarks including a large Ericsson design with 170 tests. Our tool and its modelling are made general, which is an advantage since it can be used to model the test behavior of not only scan tested systems.

The organization of the paper is as follows. An overview of related work is in Section 2 and the considered test arguments are discussed in Section 3. The system model is described in Section 4 and our integrated test scheduling, test parallelization and TAM design technique is presented in Section 5. The paper is concluded with experimental results in Section 6 and conclusions in Section 7.

2 Related Work

In order to minimize the test application time, it is desirable to schedule the tests as concurrent as possible, however, constraints and limitations must be considered. For instance, only one test can be applied at a time to each testable unit and power constraints must be carefully considered otherwise the system can be damaged.

Several test scheduling approaches have been proposed [5,1,6,7]. All are minimizing the test time but taking different issues into consideration. For instance, Chakrabarty et al. [5,6] and Muresan et al. [7] have proposed techniques to minimize test time under power limitations and conflicts. An approach designing the TAM and scheduling the tests is proposed by Cota et al. [14].

---

1. This work has been supported by the Japan Society of Promotion of Science (JSPS) under grant P01735 and the Swedish Agency for Innovation Systems (Vinnova).
3 Test Arguments

3.1 Test Power Consumption

The power consumption during testing is usually higher compared to normal operation due to that high activity is desired in testing to activate as many faults as possible and minimizing the test time. Chou et al. [6] introduced a test power model by denoting a fixed test power value to each test set. Zorian [1] and Muresan et al. [7] have used the assumption. We will also use it in our approach.

3.2 Scan-chain partitioning

The test time for scan tested cores depends on the number of assigned TAM wires. If a single TAM wire is assigned to a core, all scan chains are formed into a single chain and the test time becomes high. On the other hand, if a high number of TAM wires are assigned, the scan-chains at the core are distributed over the TAM wires and the test time is reduced.

The test power depends on the activity in the system and by disabling the scan-chains the test power can be reduced [10]. Consider an example of a core with n scan chains. If n TAM wires are assigned, all n chains are active at the same time. The test time is low but the test power consumption becomes high. On the other hand, if a single TAM wire is assigned to the core, all chains are forming a single chain, which results in a high test time. The test power consumption can be the same as in the case with several TAMs since all flip-flops in the scan chains are active. However, each of the scan chains can be loaded one at a time and that reduces the activity to 1/n [10].

3.3 Test Resource Limitations

A test source usually has a limited bandwidth. For instance, an external tester may only support a limited number of scan chains at a time [9]. The memory limitation at a test source may also put a limitation on the testing [9] and there could also be a limitation in number of available pins.

3.4 Interconnection Test

Interconnection test is used to test interconnection between cores as well as UDL placed between cores. An example of an interconnection test is illustrated in Figure 1. The test vectors are transported from the test source \( r_1 \) to the wrapped core \( C_1 \) using the TAM. The wrapper at \( C_1 \) is put in external test mode in order to perform interconnection test and the UDL, \( D_{31} \), is receiving the test vector. The test response is captured at the wrapper (also in external mode) at core \( C_2 \). In this example, a TAM is required connecting the test source \( r_1 \) and core \( C_1 \) and from core \( C_2 \) connecting test sink \( s_1 \). Surrounding cores and UDL blocks such as \( D_{32} \) in this example could interfere during an interconnection test. If so, \( D_{31} \) and \( D_{32} \) cannot be tested concurrently.

4 System Modelling

In this section, we describe our system model and the input specification to our test design tool. We illustrate the modelling and the input specification using an example. We also describe the TAM style we are assuming.

We make two observations regarding the cores in a core based environment, there is no specification:

1. given when a UDL block becomes a core,
2. saying a core must be placed in a wrapper.

In our system modelling, we assume that all parts to be tested can be partitioned into cores where some cores are placed in wrappers while other cores are not, wrapped and unwrapped. It means that the UDL blocks are seen as cores that are not placed in wrappers, i.e. they are unwrapped.

A system with a tests method for each testable part could results in a system as in Figure 1. We model such a system by extending our previous system model [2]: design with test, \( DT = (C, B, T, R_{source}, R_{sink}, P_{max}, source, sink) \), where: \( C = \{c_1, c_2, ..., c_n\} \) is a finite set of cores; each core, \( c_i \in C \), is characterized by:

\[ \{x_i, y_i\} \] placement denoted by x and y coordinates and each core consists of a finite set of blocks \( b_i = \{b_{i1}, b_{i2}, ..., b_{im}\} \) where \( m > 0 \). Each block, \( b_i \in B \), is characterized by:

- \( \min bw_{ij} \): minimal TAM bandwidth
- \( \max bw_{ij} \): maximal TAM bandwidth

Each block, \( b_{ij} \), is attached with a finite set of tests, \( b_{ij} = \{t_{ij1}, t_{ij2}, ..., t_{ijm}\} \) and each test, \( t_{ijk} \in T \), is characterized by:

- \( \tau_{ijk} \): test time
- \( p_{ijk} \): test power
- \( \text{mem}_{ijk} \): required memory for test pattern storage
- \( cl_{ijk} \): constraint list with blocks required for the test.

\( R_{source} = \{r_1, r_2, ..., r_p\} \) is a finite set of test sources where
each test source, \( t_s \in R_{\text{source}} \), is characterized by:
\[(x_i, y_i): \text{placement denoted by } x \text{ and } y \text{ coordinates}, \]
\[v_{bw_i}: \text{vector bandwidth}, \]
\[vmem_i: \text{size of vector memory}. \]

\( R_{\text{sink}} = \{s_1, s_2, ..., s_q\} \) is a finite set of test sinks; where each test sink, \( s_j \in R_{\text{sink}} \), is characterized by:
\[(x_j, y_j): \text{placement denoted by } x \text{ and } y \text{ coordinates}, \]
\[rbw_j: \text{response bandwidth}, \]

\( source: T \rightarrow R_{\text{source}} \) defines the test sources for the tests;
\( sink: T \rightarrow R_{\text{sink}} \) defines the test sinks for the tests;
\[p_{max}: \text{maximal allowed power at any time}; \]

The input specification for the example system (Figure 1) to our test design tool is outlined in Figure 2. The maximal allowed power limit is given under [Global Constraints] and at [Cores] the placement \((x,y)\) and the blocks at each core are specified. The placement \((x,y)\) of each test source, its possible bandwidth, and available memory are given at [Generators]. At [Evaluators], the placement \((x,y)\) and the maximal allowed bandwidth for each test sink is given. For each of the tests the following is specified under [Tests], the test name, test power, test time, test source (test pattern generator), test sink (test response evaluator), minimal and maximal bandwidth, memory requirement and optional interconnection test with another core. For instance, test \( t_2 \) is an interconnection test between core \( c_1 \) and core \( c_2 \) (Figure 2). The tests for each block are specified at [Blocks] and under [Constraints], the blocks required in order to apply a test are listed. Note, the possibility to specify idle power for each block, which is implemented but to simplify the discussion it is excluded from the system model above.

The advantages of this model are that we can model a system with wide range tests (scan tests and non-scan tests such as delay, timing and cross-talk tests) and constraints, for instance we can model:

- interconnection test for UDL (unwrapped cores),
- any combination of test resources, for instance a test source can be on-chip while the test sink is off-chip and vice versa,
- any number of tests per block (testable unit),
- memory requirements at test sources,
- bandwidth limitations at test resources,
- constraints among blocks, which allows modelling of constraints such as cores embedded in cores.

Initially, no TAM exists in the system. Our technique adds a set of TAMs, \( tam_1, tam_2, ..., tam_n \) where each \( tam_i \) is a set of wires with bandwidth \( n_i \). We assume that we can partition the TAM connected to a set of cores freely, which means we are not limited to assigning all wires in a TAM to one core at a time or dedicate TAM wires to a single core.

The TAM is modelled as a directed graph, \( G=(N, A) \), where a node \( n_i \) in \( N \) corresponds to a member of \( C \) or \( R_{\text{source}} \) or \( R_{\text{sink}} \). An arc, \( a_{ij} \in A \), between two nodes \( n_i \) and \( n_j \) indicate the existence of a wire and a wire \( w_{ij} \) is a set of arcs, for instance a wire \( w_{ij} \) from \( c_1 \) to \( c_3 \) passing \( c_2 \) is given by the two arcs: \( \{a_{12}, a_{23}\} \).

### Example design

<table>
<thead>
<tr>
<th>[Global Constraints]</th>
<th>MaxPower = 25</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Cores] name</td>
<td>( x ) ( y ) ( \text{(block1, block2, ..., block n)} )</td>
</tr>
<tr>
<td>( c_1 )</td>
<td>10 30 ( {b11} )</td>
</tr>
<tr>
<td>( c_2 )</td>
<td>10 20 ( {b21, b22} )</td>
</tr>
<tr>
<td>( c_3 )</td>
<td>20 30 ( {b31, b32} )</td>
</tr>
<tr>
<td>[Generators] name</td>
<td>( x ) ( y ) maxbw memory</td>
</tr>
<tr>
<td>( r_1 )</td>
<td>10 40 3 ( 100 )</td>
</tr>
<tr>
<td>( r_2 )</td>
<td>10 10 1 ( 100 )</td>
</tr>
<tr>
<td>[Evaluators] name</td>
<td>( x ) ( y ) maxbw</td>
</tr>
<tr>
<td>( s_1 )</td>
<td>20 40 4</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>20 10 4</td>
</tr>
<tr>
<td>[Tests] name</td>
<td>pwr time tpg tre minbw maxbw mem</td>
</tr>
<tr>
<td>( t_1 )</td>
<td>10 15 r1 s1 1 2 10 no</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>5 10 r1 s2 2 4 5 c2</td>
</tr>
<tr>
<td>[Blocks] #Syntex:</td>
<td>name idle pwr ( {\text{test1, test2, ..., test n}} )</td>
</tr>
<tr>
<td>( b_{11} )</td>
<td>1 ( {t1, t2, t3} )</td>
</tr>
<tr>
<td>( b_{21} )</td>
<td>1 ( {t4, t5} )</td>
</tr>
<tr>
<td>[Constraints] Syntax: ( \text{test (block1, block2, ..., block n)} )</td>
<td></td>
</tr>
<tr>
<td>( t_1 )</td>
<td>( {b_{11}} )</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>( {b_{11}, b_{21}, b_{22}, b_{31}, b_{32}} )</td>
</tr>
<tr>
<td>// constraint for all tests in similar way</td>
<td>( t_{12} ) ( {b_{11}} )</td>
</tr>
</tbody>
</table>

Figure 2. Test specification of the system in Figure 1.

To model the assignment of cores to TAM wires connecting a test source, \( n_{source} \), a set of cores, \( n_1, n_2, ..., n_n \), and a test sink, \( n_{sink} \), we use:

\[ n_{source} \rightarrow [n_1, n_2, ..., n_n] \rightarrow n_{sink} \]

where \( [\ ] \) indicates that the nodes (cores) are included (assigned) in this TAM but not ordered.

The length, \( l_j \), of a test wire, \( w_j \), is given by:

\[ adist(n_j, n_i) + \sum_{k=2}^{n} adist(n_{k-1}, n_k) + adist(n_{n}, n_1) \]

where \( n_i = n_{source} \) and \( n_n = n_{sink} \) and the function \( adist \) gives the absolute distance between two nodes, i.e.:

\[ adist(n_j, n_i) = |x_j - x_i| + |y_j - y_i| \]

A set of wires form a \( tam_i \) and the routing cost is:

\[ tam_{length_i} = l_i \times bandwidth_i \]

and the total TAM routing cost in the system is given by:

\[ c_{tam} = \sum_{i=1}^{n} tam_{length_i} \]

The total cost for a test solution is given by:

\[ \alpha \times \text{test time} + \beta \times c_{tam} \]

where \( \text{test time} \) is the total test application time, \( c_{tam} \) (defined above) is the total wiring cost, and \( \alpha \) and \( \beta \) are designer specified constants determining the relative importance of the test time and the TAM cost.

### 5 Our Approach

In this section we describe our approach to integrate test scheduling, scan-chain partitioning and TAM design. For a
given floor-planned system with tests, modelled as in Section 4, we have to:

- determine the start time for all tests,
- determine the bandwidth for each test,
- assign each test to TAM wires,
- determine the number of TAMs,
- determine the bandwidth of each TAM, and
- route each TAM,

while minimizing the test time and the TAM cost considering constraints and power limitations. Compared to our previous approach [2,3,4] the following summarizes the improvements:

- **Test scheduling.** In [2] when a test was selected and all constraints were satisfied, a TAM was designed. The approach always minimized test time at the expense of the TAM cost. In this approach, a cost function including test time and TAM cost guides our algorithm.
- **Scan chain partitioning (test parallelization).** In [4] we maximized the bandwidth for each test, which resulted in a low test time, however, the draw-back is a higher TAM cost. Now, a cost function guides our algorithm.
- **TAM design.** In [2] when a test was selected and a free TAM existed, it was selected. If an extension was required, an extension was made minimizing the additional TAM. A disadvantage of the approach is illustrated in Figure 3 where a test D is to be connected using the dashed line. A re-routing as A,C,D,B would include D at no additional cost.

The cost function guiding our algorithm is:

\[ c = f_{\text{tam}} \times \text{tlength} + f_{\text{time}} \times \tau_{\text{start}} \]

where: \( f_{\text{tam}} \) is a test using \( \text{tam} \) and \( \text{tlength} \) is the cost of the tam wiring (Eq. 4). \( \tau_{\text{start}} \) is the time when \( t_{ij} \) can start, and the designer specified factors \( f_{\text{tam}} \) and \( f_{\text{time}} \) are used to set the relative importance between test time and TAM cost.

### 5.1 Bandwidth Assignment

Scan chain partitioning allows a flexible bandwidth assignment for each test depending on the bandwidth limitations at the block under test and the bandwidth limitations at the test resources.

The test time (see Section 3.2) for a test \( t_{ij} \) at block \( b_{ij} \) at core \( c_i \) is given by:

\[ \tau'_{ij} = \left[ \tau_{ij}/bw_{ij} \right] \]

and the test power:

\[ p'_{ij} = p_{ij} \times bw_{ij} \]

where \( bw_{ij} \) is the bandwidth [4].

Note, that the test times can often be determined very precisely, however, other issues such as power consumption is not given at such accuracy; justifying our estimations.

Combining the TAM cost and the test time (Equation 7), we get for each block \( b_{ij} \) and its tests \( t_{ijk} \):

\[ \text{cost}(b_{ij}) = \sum_{k} \left[ f_{\text{tam}} \times \text{tlength} + f_{\text{time}} \times \tau_{ijk}/bw_{ij} \right] \times f_{\text{time}} \]

where: \( l_i = \text{source}(t_{ijk}) \rightarrow c_i \rightarrow \text{sink}(t_{ijk}) \) and \( k \) is the index of all tests at the block. To find the minimum cost of Equation 9, the derivative in respect to \( bw \). Naturally, when selecting the \( bw_{ij} \), we also consider the bandwidth limitations at each block.

### 5.2 Test scheduling

Our test scheduling algorithm is outlined in Figure 4. First, the bandwidth is determined for all blocks (Section 5.1) and the tests are sorted based on a key (time, power or test weight). The outermost loop terminates when all tests are scheduled. In the inner loop, the first test is picked and after calling the create tamplan (Section 5.3) and based on the cost function a TAM is selected or designed for the test. If the TAM factor is important, a test can be delayed in order to use an existing TAM. If all constraints are satisfied, the test is scheduled and the TAM assignment is executed. Finally, all TAMs are optimized.

### 5.3 TAM Planning

In the TAM planning phase, we:

- create the TAMs,
- determine the bandwidth of each TAM,
- assign tests to the TAMs, and
- determine the start time for each test.

The difference compared to our previous approach is that in the planning phase we only determine the existence of the TAMs but not their routing.

For a selected test, the cost function is used (create tamplan(\( t' \), test) Figure 5) and if a test is to be scheduled, the time (\( t' \)) when the test is to start and its TAM are determined. If all constraints are satisfied, the TAM plan is determined (execute(tamplan)) (Figure 6).

To compute the cost of extending a TAM wire with a node, the length of required additional wire length is computed. Since the order on a TAM is not decided, we need an estimation technique and for most TAMs, the

for all blocks bandwidth = bandwidth(block)

sort the tests descending based on time, power or test weight

until all tests are scheduled

\[ \text{tamplan} = \text{create tamplan}(t', \text{test}) \] // see Figure 5 //

\[ t' = t + \text{delay(tamplan)} \]

if all constraints are satisfied

\[ \text{schedule}(t') \]

\[ \text{execute(tamplan)} \] // see Figure 6 //

remove test from list

\[ t = \text{first time the next test can be scheduled} \]

order (tam) // see Figure 8 //

Figure 3. Illustration of our previous TAM design.

Figure 4. Test scheduling algorithm.
2. The estimated length, cost according to:
   \[ L' = \min(L_{eijk} + \text{delay}(t_{ijk}) \times f_{time}) \]

   Using this cost function, we get a trade-off between

   \[ \text{cost} = \text{cost function}(	ext{tam cost}, \text{time}, \text{test}) \]

   for all tams connecting the test source and test sink used by the test,
   select the one with lowest total cost

   \[ \text{tam cost} = 0; \]
   \[ \text{demanded bandwidth} = \text{bandwidth(test)} \]
   \[ \text{if bandwidth(test)>max bandwidth selected tam} \]
   \[ \text{demanded bandwidth}=\max \text{bandwidth(tam)} \]
   \[ \text{tam cost} = \text{tam cost} + \text{cost for increasing bandwidth of tam;} \]
   \[ \text{time}=\text{first free time (demanded bandwidth)} \]
   \[ \text{sort tams ascending according to extension (t, test)} \]
   \[ \text{while more demanded bandwidth} \]
   \[ \text{tam}=\text{next tam wire in this tam;} \]
   \[ \text{tam cost} = \text{tam cost} + \text{cost (bus, demanded bandwidth)} \]
   \[ \text{update demanded bandwidth accordingly;} \]
   \[ \text{total cost}=\text{cost function (tam cost, time, test);} \]

1. The nodes \( N \) (test sources, test sinks and cores) are evenly distributed over the area, \( A = \text{width} \times \text{height} \), and \( N_\text{x} \times N_\text{y} \times \Delta^2 \) is the number of cores on the \( x \) and \( y \) axis, respectively:

   \[ \Delta = \sqrt{A / N_\text{x} \times N_\text{y}} \]

2. The estimated length, \( e_{ijk} \), of a wire, \( w_{ij} \), with \( k \) nodes is:

   \[ e_{ij} = \max_{s,y,k} [l(n_{source} \rightarrow n_j \rightarrow n_{sink}), \Delta \times (k + 1)] \]

   It means that we compute the maximum between the length as the longest created wire and the sum of the average distances for all needed arcs (wire parts). An example, let \( n_{furthest} \) be the node creating the longest wire, and \( n_{new} \) the node to be added, the estimated wiring length after inserting \( n_{new} \) is given by (Eq. 11):

   \[ e_{ij}' = \max \left[ \min \left[ l(n_{source} \rightarrow n_{new} \rightarrow n_{furthest} \rightarrow n_{sink}) \right], \Delta \times (k + 2) \right] \]

   For a TAM, the extension is given as the summation of all extensions of the wires included in the TAM that are needed in order to achieve required bandwidth. The TAM selection for a test \( t_{ijk} \) is based on the TAM with the lowest cost according to:

   \[ \text{cost} = (e_{ij}' - e_{ij}) \times f_{tam} + \text{delay}(\text{tam}_{ij}, t_{ijk}) \times f_{time}. \]

5.5 TAM Optimization

Above we created the TAMs for the system, assigned each test to a TAM and determined the bandwidth. In this section, we determine the routing of each of the TAMs, \( \text{order(tam)} \) (Figure 4) and it is based on a simplification of an algorithm presented by Caseau and Laburthe [13]. The notation \( \text{TG} \rightarrow \text{[A, D]} \rightarrow \text{SA} \) was used to indicated that core A and D were assigned to the same TAM, however, the order in \( \text{[A, D]} \) was not determined (Equation 1), which is the objective in this section. We use:

\[ n_{source} \rightarrow n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow n_n \rightarrow n_{sink} \]

with a new TAM and delaying a test on an existing TAM. For a newly created TAM, the delay for a test is 0 (since no other test is scheduled on the TAM and the test can start at time 0); \( \text{new cost}(t_{ijk}) = l(n_{source} \rightarrow n_j \rightarrow n_{sink}) \times f_{tam}. \)
until all nodes are removed, resulting in a TAM: 2. D is inserted satisfying Eq. 16. The algorithm continues
connection is added between TG and SA. At step 1, in the
2. At step 0, the nodes are ordered, C,D,A,B, and a
add direct after the position where Eq. 14 is satisfied
remove first node from list and insert in the final list
When referring to our technique, unless stated, the
reported results are from using sorting based on an initial
sorting of the test based on the key: tsp and our previous
techniques are referred to as SA (Simulated Annealing) [3]
and DATE [2]. For our approach a factor \( f_{tam}:f_{time} \) is
given to guide our algorithm depending on the importance of
test time and TAM cost (see Section 5). The final cost of a
test solution is evaluated using: \( Time = \alpha \times test \ time + \beta \times TAM \ cost \) where
\( \alpha \) and \( \beta \) are designer specified constants determining the
relative importance of the test time and the TAM cost (we
use \( \alpha = 1 \) and \( \beta = 1 \) unless stated). SA also uses the function.
For the experiments we have used a Pentium II 350 MHz
processor with 128 Mb RAM. Our previous results referred
to as DATE [2] and SA [3] were performed on a Sun Ultra
Sparc 10 with 450 MHz processor and 256 Mb RAM [2,3].
To simplify the comparison we assume the two to be equal.

### 6.1 Test Scheduling

We have made experiments comparing our test scheduling
technique with previously proposed techniques, Table 3.
For instance, the optimal solution on design Muresan is 25
time units, which is found by SA after 90 seconds while our
technique finds it within 1 second.

### 6.2 Test Scheduling and TAM design

The results on integrated test scheduling and TAM design are
collected in Table 4. On ASIC Z when not considering
idle power, our technique with (2:1) produces a result 2.4%
from the result produced by SA within 1 second while SA
required 865 seconds.
For the experiments on the Ericsson design we have used $\alpha=1$ and $\beta=3100$. In our approach finds the optimal test time, which is also found by SA and DATE. However, the results using SA and DATE do not support TAM of higher bandwidth than 1 and therefore only test time is reported (the experiments were performed ignoring TAM design). In Our* we have forced the bandwidth to 1. The lowest total cost is found for Our* (1:50). For the test time and TAM cost, when the former increases, the latter decreases, which is expected.

### 7 Conclusions

Test time minimization and efficient TAM design is becoming important due to the increasing amount of test data to be transported in a SOC design. We have proposed an integrated technique for test scheduling, scan chain partitioning and TAM design, which minimizes test time and TAM cost while considering test conflicts and power limitations. In our approach it is possible to model a variety of tests as well as tests of wrapped cores, unwrapped cores and user-defined logic. We have implemented the technique and made several experiments.

### References


For System S (\(\alpha=1\) and \(\beta=3100\)) we have made two experiments combining test scheduling, test parallelization and TAM design where the results indicate the usefulness (cost reduction in the range from 9% to 12%).

For System L we have used \(\alpha=1\) and \(\beta=30\) and both SA and DATE do not support TAM of higher bandwidth than 1 and therefore only test time is reported. For System L we have performed experiments combining test scheduling, test parallelization and TAM design, which minimizes test time and TAM cost while considering test conflicts and power limitations. In our approach it is possible to model a variety of tests as well as tests of wrapped cores, unwrapped cores and user-defined logic. We have implemented the technique and made several experiments.

### 6.3 Scheduling, Parallelization and TAM design

We have performed experiments combining test scheduling, TAM design and test parallelization, Table 4.

<table>
<thead>
<tr>
<th>Step</th>
<th>Selected</th>
<th>Order before selection</th>
<th>Length of each TAM partition</th>
<th>Order after selection</th>
<th>TAM length</th>
<th>Remaining in list</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>TG→SA</td>
<td>0</td>
<td>TG→SA</td>
<td>30</td>
<td>C,D,A,B</td>
</tr>
<tr>
<td>1</td>
<td>D</td>
<td>TG→C→SA</td>
<td>18.2</td>
<td>TG→C→S</td>
<td>14+22=36</td>
<td>D,A,B</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>TG→C→D→SA</td>
<td>6,14,20</td>
<td>TG→C→D→SA</td>
<td>10+10+14+4=44</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>TG→C→A→D→B→SA</td>
<td>20,14,14,14,6</td>
<td>TG→A→C→D→B→SA</td>
<td>10+10+10+10+5=50</td>
<td>-</td>
</tr>
</tbody>
</table>

For the extended ASIC Z, the total cost using our technique is in the range from 8% to 22% better than SA and the results computed using our technique are made within 1 second compared to 4549 seconds using SA. The results here also demonstrate the improvement regarding the TAM cost using our technique.

For the experiments on the Ericsson design we have used $\alpha=1$ and $\beta=2$. In the previous approaches no bandwidth limitations were given on the external tester [2,3], however, here we assume a bandwidth limitation of 12. Our technique (15:1) produces better results than SA both in respect to test time (6.6%) and TAM (13.4%) leading to a total cost (8.6%) of (15:1) produces better results than SA both in respect to test time (6.6%) and TAM (13.4%) leading to a total cost (8.6%) of 1. Our technique (15:1) produces better results than SA both in respect to test time (6.6%) and TAM (13.4%) leading to a total cost (8.6%) with a variety of test scheduling, test parallelization and TAM design where the results indicate the usefulness (cost reduction in the range from 9% to 12%).

For System S we have used \(\alpha=1\) and \(\beta=3100\) and in our previous approach we assumed that the external tester supported several tests at the same time [3]. For System S, we have used \(\alpha=1\) and \(\beta=3100\) and in our previous approach we assumed that the external tester supported several tests at the same time [3]. For System S, we have used \(\alpha=1\) and \(\beta=3100\) and in our previous approach we assumed that the external tester supported several tests at the same time [3]. For System S, we have used \(\alpha=1\) and \(\beta=3100\) and in our previous approach we assumed that the external tester supported several tests at the same time [3]. For System S, we have used \(\alpha=1\) and \(\beta=3100\) and in our previous approach we assumed that the external tester supported several tests at the same time [3]. For System S, we have used \(\alpha=1\) and \(\beta=3100\) and in our previous approach we assumed that the external tester supported several tests at the same time [3].
Table 4. Experimental results on (1)integrated test scheduling and TAM design and on (2) combined test scheduling, TAM design and test parallelization. For results on System L with index 1 we compare with Our (1:15500).

<table>
<thead>
<tr>
<th>Approach Design</th>
<th>Test Application Time</th>
<th>Test Access Mechanism</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Test time (τ)</td>
<td>Diff. to SA TAM cost (tam)</td>
<td>Difference to SA (%)</td>
</tr>
<tr>
<td>ASIC Z</td>
<td>326</td>
<td>180</td>
<td>506</td>
</tr>
<tr>
<td>DATE[2]</td>
<td>262</td>
<td>-19.6%</td>
<td>300</td>
</tr>
<tr>
<td>Our (1:1)</td>
<td>262</td>
<td>-19.6%</td>
<td>280</td>
</tr>
<tr>
<td>Our (2:1)</td>
<td>338</td>
<td>3.7%</td>
<td>180</td>
</tr>
<tr>
<td>Our (10:1)</td>
<td>395</td>
<td>21.2%</td>
<td>160</td>
</tr>
<tr>
<td>SA [3]</td>
<td>270</td>
<td>-</td>
<td>560</td>
</tr>
<tr>
<td>DATE[2]</td>
<td>287</td>
<td>6.3%</td>
<td>660</td>
</tr>
<tr>
<td>Our (1:1)</td>
<td>262</td>
<td>-3.0%</td>
<td>500</td>
</tr>
<tr>
<td>Our (1:1)</td>
<td>264</td>
<td>-2.2%</td>
<td>480</td>
</tr>
<tr>
<td>Our (5:2)</td>
<td>266</td>
<td>-1.5%</td>
<td>380</td>
</tr>
<tr>
<td>Our (7:1)</td>
<td>305</td>
<td>13.0%</td>
<td>360</td>
</tr>
<tr>
<td>Our (10:1)</td>
<td>315</td>
<td>16.7%</td>
<td>340</td>
</tr>
<tr>
<td>Our (20:1)</td>
<td>477</td>
<td>76.7%</td>
<td>240</td>
</tr>
<tr>
<td>SA [3]</td>
<td>33082</td>
<td>-</td>
<td>6910</td>
</tr>
<tr>
<td>DATE[2]</td>
<td>34762</td>
<td>5.1%</td>
<td>8520</td>
</tr>
<tr>
<td>Our (15:1)</td>
<td>30899</td>
<td>-6.6%</td>
<td>5985</td>
</tr>
<tr>
<td>Our (120:1)</td>
<td>45060</td>
<td>36.2%</td>
<td>5335</td>
</tr>
<tr>
<td>Our (180:1)</td>
<td>60437</td>
<td>82.7%</td>
<td>4855</td>
</tr>
<tr>
<td>Our (300:1)</td>
<td>75457</td>
<td>128%</td>
<td>4405</td>
</tr>
<tr>
<td>Our (400:1)</td>
<td>98273</td>
<td>297%</td>
<td>3985</td>
</tr>
<tr>
<td>Our (1:0)</td>
<td>235084</td>
<td>611%</td>
<td>3175</td>
</tr>
<tr>
<td>Ericsson</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE[2]</td>
<td>396194</td>
<td>0%</td>
<td>160</td>
</tr>
<tr>
<td>Our (15000:1)</td>
<td>996194</td>
<td>0%</td>
<td>320</td>
</tr>
<tr>
<td>Our (25000:1)</td>
<td>1063930</td>
<td>6.8%</td>
<td>160</td>
</tr>
<tr>
<td>Our (45000:1)</td>
<td>1148410</td>
<td>15.3%</td>
<td>140</td>
</tr>
<tr>
<td>Our (65000:1)</td>
<td>1152180</td>
<td>15.7%</td>
<td>100</td>
</tr>
<tr>
<td>SA [3]</td>
<td>506730</td>
<td>-49.1%</td>
<td>260</td>
</tr>
<tr>
<td>DATE[2]</td>
<td>426912</td>
<td>-57.1%</td>
<td>300</td>
</tr>
<tr>
<td>System S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE[2]</td>
<td>996194</td>
<td>0%</td>
<td>160</td>
</tr>
<tr>
<td>Our (1:15500)</td>
<td>996194</td>
<td>0%</td>
<td>320</td>
</tr>
<tr>
<td>Our (1:3000)</td>
<td>318</td>
<td>0.6%</td>
<td>9490</td>
</tr>
<tr>
<td>Our (1:500)</td>
<td>322</td>
<td>1.9%</td>
<td>5140</td>
</tr>
<tr>
<td>Our (1:100)</td>
<td>343</td>
<td>8.5%</td>
<td>2420</td>
</tr>
<tr>
<td>Our (1:50)</td>
<td>360</td>
<td>13.9%</td>
<td>1750</td>
</tr>
<tr>
<td>Our (1:10)</td>
<td>399</td>
<td>26.3%</td>
<td>1030</td>
</tr>
<tr>
<td>Our (1:5)</td>
<td>463</td>
<td>46.5%</td>
<td>710</td>
</tr>
<tr>
<td>Our (1:2)</td>
<td>593</td>
<td>87.7%</td>
<td>510</td>
</tr>
<tr>
<td>Our (1:1)</td>
<td>710</td>
<td>125%</td>
<td>490</td>
</tr>
<tr>
<td>SA [3]</td>
<td>426912</td>
<td>-57.1%</td>
<td>300</td>
</tr>
</tbody>
</table>


