Wrapper and TAM Co-Optimization for Reuse of SoC Functional Interconnects

Tomokazu Yoneda† and Hideo Fujiwara†
†Graduate School of Information Science, Nara Institute of Science and Technology
Kansai Science City, 630-0192, Japan
{yoneda, fujiwara}@is.naist.jp

Abstract
This paper presents a wrapper and TAM co-optimization method for reuse of SoC functional interconnects to minimize test time under area constraint. The proposed method consists of (1) an ILP formulation for wrapper and transparent TAM co-optimization, and (2) a simulated annealing based heuristic approach to reduce the computational cost of the proposed ILP model. Experimental results show the effectiveness of the proposed methods compared to the previous transparency-based TAM approaches and the conventional dedicated test bus approaches.

1 Introduction
SoCs are increasingly designed and tested in a modular fashion [1], and the following three are key components for the modular test: (1) wrapper, (2) test access mechanism (TAM) and (3) test scheduling. A number of approaches have been proposed for wrapper and TAM design including test scheduling problem [2, 3, 4, 5]. These approaches use the infrastructure dedicated to test as TAMs. However, regardless of how efficient the wrapper, TAM and test schedule optimization are, the TAM dedicated to test requires considerable area overhead.

Therefore, a number of approaches have been proposed for the TAM architectures which are not dedicated to test, but reuse the existing components in the SoC. They are roughly classified into three types: 1) the method reusing functional buses [6, 7], 2) the methods reusing functional networks [8, 9] and 3) the methods based on transparency [10, 11, 12, 13]. The wrapper and TAM co-optimization problem to minimize test time was discussed in the methods reusing functional buses and networks while there is no discussion on it in the methods based on transparency. This is because (1) they didn’t consider scan design explicitly and (2) it is potentially difficult to achieve concurrent test by the transparency-based TAMs (more discussion in Section 2).

To the best of our knowledge, this paper presents a wrapper and transparent TAM co-optimization method to minimize test time under area constraint for the first time. We present an integer linear programming (ILP) formulation for the wrapper and transparent TAM co-optimization. Though the proposed ILP model is effective for small SoCs, we cannot find the optimal solution within reasonable time for large SoCs. Therefore, we also propose a simulated annealing (SA) based heuristic approach to reduce the computational cost of the proposed ILP model. Experimental results show the effectiveness of the proposed methods compared to the previous transparency-based TAM approaches and the conventional dedicated test bus approaches.

2 Motivation
In this section, we discuss the limitations of the previous transparency-based TAM approaches and present an effective wrapper configuration for transparent TAMs.

Fig. 1(a) and (b) show an example SoC that the transparency-based methods target and its transparent TAM for core C2, respectively. In the previous methods based on transparency, they provided complete transparent access for every functional port shown in Fig. 1(b). Consequently, C2 cannot be tested concurrently with the other cores, and only the sequential test is possible. On the other hand, if we use the IEEE 1500 wrapper [14] to test the core, we can select any bit-width of transparent access to test the core in the similar way to the dedicated TAM approaches. Fig. 1(c) shows an example of 3-bit transparent test access to C2. However, C2 still cannot be tested concurrently with the other cores even though C1 is free. This is because only the way to propagate the test responses of C1 is to pass through C2 in the transparency-based TAM design in this example.

In this paper, we introduce a wrapper configuration that can perform INTEST and BYPASS modes simultaneously in order to increase test concurrency in the transparent TAM design effectively. Fig. 2(a) shows an example of the proposed wrapper configuration where INTEST with 3-bit functional TAM and BYPASS for 2-bit transparency are realized concurrently. Bypass registers and multiplexers are added not to prevent the core from being tested. By using the proposed wrapper configurations effectively in the transparency-based TAM design, we can increase test concurrency and reduce the overall test time while keeping the area overhead low. For example, we can test C1 and C2 concurrently without increasing the SoC functional interconnects by using the proposed wrapper configuration for C2 as shown in Fig. 2(b).

3 Problem Formulation
In the dedicated test bus based TAM designs, it is well known that there is a trade-off relation between TAM area and test time.

Figure 1: (a) An example system S1, (b) Transparent test access for core C2, (c) Transparent test access for C2 with 3-bit wrapper.

†This work was supported in part by Japan Society for the Promotion of Science (JSPS) under Grants-in-Aid for Young Scientists(B)(No.18700046).
We observe the similar trend in the transparent TAM approach using the proposed wrapper configurations. We can reduce test time by increasing the bit-width of SoC functional interconnects and SoC external I/O ports. However, we consider that the cost for adding extra SoC I/O ports is much higher than that for increasing internal interconnects. Therefore, we consider the following optimization problem in this paper.

**Definition 1** $P_{opt}$: Given a set of cores with test parameters, a set of interconnects and maximum allowable increase of interconnects in bit $C_{max}$, determine a wrapper and transparency-based TAM for each core such that: (1) the bit-width of each SoC I/O port and its associated interconnect remains the same, (2) the total increase of interconnects does not exceed $C_{max}$, and (3) the overall test time is minimized.

4 Wrapper and Transparent TAM Co-Optimization

4.1 ILP Formulation

In this paper, we use the session based test scheduling where test sets are grouped into sessions and new tests are allowed to start only when all tests in the preceding session are completely executed. To solve $P_{opt}$, for each core $k$, we decide a test session and TAM width to test $k$, and select interconnects used as TAM for $k$. For each interconnect, the sum of the TAM width used to test cores scheduled in a session is the final TAM width used as TAM in the session. If the final TAM width exceeds the original bit-width of the interconnect, we have to increase the bit-width of it. From the above decisions and selections, finally, we can determine a wrapper configuration for each core in each session to provide the required functionality. We present an ILP formulation to solve $P_{opt}$ as follows.

**0-1 Variables**:

\[ s_{j,k} : s_{j,k} = 1 \text{ if core } k \text{ is tested in session } j \]

\[ r_{j,k,l} : r_{j,k,l} = 1 \text{ if core } k \text{ is tested in session } j \text{ with } l \text{ bit TAM} \]

**Integer Variables**:

\[ x_{i,k,j} : \text{TAM width on interconnect } i \text{ to test core } k \text{ in session } j \]

**Other Notations**:

\[ E_{in,k} : \text{a set of input interconnects of core } k \]

\[ E_{out,k} : \text{a set of output interconnects of core } k \]

\[ w(i) : \text{the original bit-width of interconnect } i \]

**time($k, l$) :** the test time of core $k$ with $l$ bit TAM

**Constraints**:

1. $\sum_j s_{j,k} = 1$ for $\forall k$, i.e., every core is scheduled in exactly one session

2. $\sum_l r_{j,k,l} = s_{j,k}$ for $\forall j, k$, i.e., every core under test $k$ has exactly one wrapper configuration for INTEST

3. $\sum_{j \in E_{in,k}} x_{i,j,k} = \sum_{j \in E_{out,k}} x_{o,j,k} = \sum_l l \cdot r_{j,k,l}$ for $\forall j, k$, i.e., every core under test $k$ has a set of input/output interconnects used as input/output TAM with the width corresponding to the selected wrapper configuration

4. $\sum_{j \in E_{in,k}} x_{i,j,k} = \sum_{j \in E_{out,k}} x_{o,j,k}$ for $\forall j, k(k^{'\neq} k)$, i.e., if core $k'$ (core under test $k$) is used as a part of TAM for $k$, the sum of the TAM width for $k$ at the input ports of $k'$ is equal to the sum of the TAM width for $k$ at the output ports of $k'$ (TAM width preservation)

5. $\max_j(\sum_j x_{i,j,k}) \leq w(i)$ for $\forall i$ associated with SoC I/O ports, i.e., every interconnect associated with an SoC I/O port cannot be increased

6. $C_{max} \geq \sum_k \max_i(\sum_j x_{i,j,k} + w(i) - w(i))$, i.e., the total increase of internal interconnect does not exceed $C_{max}$

**Objective**:

Minimize $\sum_j \max_i(\sum_l \text{time}(k, l) \cdot r_{j,k,l})$

We can easily include the dedicated test bus design in the proposed ILP formulation and consider transparency-based TAM design, dedicated test bus based TAM design and their hybrid TAM design in the proposed ILP formulation.

4.2 Experimental Results for ILP

We made experiments on three SoCs: $S_1$, we handcrafted shown in Fig. 1(a), $d695$ and $p93791$ from ITC’02 SOC Test Benchmarks [15]. The test parameters for core $C_1$, $C_2$, $C_3$ and $C_4$ in $S_1$ are identical to module 1, 5, 6 and 10 in $p93791$, respectively. For $d695$ and $p93791$, since the original benchmark SoCs do not have any data on the connectivity between cores, we used randomly-generated interconnects for the SoCs. Table 1 shows the characteristics for the three SoCs. Column 4 denotes the number of SoC I/O bits which can be used as the transparent TAM I/O. Column 5 denotes the number of the dedicated test bus based TAM I/O bits added to the original SoC for comparison purpose.

Table 2, 3 and 4 show the test time results for $S_1$, $d695$ and $p93791$, respectively. Columns “tTAM”, “dTAM” and “tTAM+dTAM” denote the cases where we design the transparent TAM only, the dedicated test buses based TAM only, and their hybrid TAM, respectively. We used a commercial ILP solver ILOG CPLEX [16] on a SunFireV490 workstation with UltraSPARC IV+ 1.8 GHz processor and 32 GB memory for all the experiments. We set the time limit of the ILP solver to 10, 600 and 7200 seconds for $S_1$, $d695$ and $p93791$, respectively. The bold number in the tables means that the ILP solver can reach the optimal solution within the time limit. On the other hand, the non-bold number denotes the best intermediate solution at the time limit. “*” and “NA” mean that the ILP solver cannot find any intermediate solution.

<table>
<thead>
<tr>
<th>Table 1: Characteristics for three SoCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoC</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>$S_1$</td>
</tr>
<tr>
<td>$d695$</td>
</tr>
<tr>
<td>$p93791$</td>
</tr>
</tbody>
</table>
the dedicated TAM design when transparent TAM design can achieve exactly the same test time as the optimal solution in all cases within 10 seconds. Second, the have the following four observations. First, the ILP solver can get in some cases for get the optimal solution in many cases for ILP in almost all cases. However, we cannot find any intermediate solution. These results motivated us to present an effective heuristic approach based on simulated annealing for large SoCs. The SA algorithm is shown in Fig. 3. The SA algorithm starts to generate an ILP model described in the previous section and solve the ILP model with a randomly generated initial session assignment. Then a neighboring session assignment is randomly created from the current session assignment. If the test time of the neighboring assignment be better than the current assignment, the neighboring assignment is accepted. If the test time of the neighboring assignment is not better than the current assignment, it can be accepted at a certain probability which is a function of a parameter referred as temperature. During the optimization process, the temperature is decreased and there is a lower probability of accepting an inferior solution. The optimization process terminates when the temperature reaches the given stop criteria.

5.2 Experimental Results for SA

We set the parameters in the proposed SA algorithm so that the computation time for d695 and p93791 become 60 and 1800 seconds, respectively. Tables 5 and 6 show the test time results for d695 and p93791, respectively. Columns 2, 5 and 8 (i.e., “10TS+ILP” in Table 5 and “32TS+ILP” in Table 6) denote the test time given by the original ILP model proposed in Section 3. Columns 3, 6 and 9 (“5TS+SA”), denote the test time given by the SA algorithm with 5 test sessions. The number in parentheses denotes the relative difference from the original ILP model proposed in Section 3.

From the results for d695 shown in Table 5, we observe that the ILP solver can provide slightly better results by limiting the number of test sessions to 5 in almost all cases. However, it still cannot get the optimal solution for many cases within the given time limit. On the other hand, the proposed SA based approach can achieve approximately the same test time as “5TS+ILP” with 10 times shorter computational time.

For p93791 shown in Table 6, we can get 10 to 36% reduction in test time by limiting the number of test session from 32 to 5.

Table 2: Test time results (#cycles) for S₁.

<table>
<thead>
<tr>
<th>Cmax</th>
<th>tTAM (16bit IO)</th>
<th>dTAM (16bit IO)</th>
<th>tTAM+dTAM (32bit IO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>oo</td>
<td>6020325</td>
<td>95466</td>
<td>341856</td>
</tr>
<tr>
<td>25</td>
<td>6020325</td>
<td>95466</td>
<td>349288</td>
</tr>
<tr>
<td>12</td>
<td>6020325</td>
<td>2658613</td>
<td>600728</td>
</tr>
<tr>
<td>8</td>
<td>774299</td>
<td>5317087</td>
<td>774299</td>
</tr>
<tr>
<td>4</td>
<td>1112067</td>
<td>NA</td>
<td>1112067</td>
</tr>
<tr>
<td>0</td>
<td>1978000</td>
<td>NA</td>
<td>1978000</td>
</tr>
</tbody>
</table>

CPU/sec

10
10
10

Table 3: Test time results (#cycles) for d695.

<table>
<thead>
<tr>
<th>Cmax</th>
<th>tTAM (14bit IO)</th>
<th>dTAM (16bit IO)</th>
<th>tTAM+dTAM (32bit IO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>oo</td>
<td>50383</td>
<td>24201</td>
<td>15730</td>
</tr>
<tr>
<td>100</td>
<td>50383</td>
<td>120188</td>
<td>29763</td>
</tr>
<tr>
<td>25</td>
<td>50383</td>
<td>91874</td>
<td>32955</td>
</tr>
<tr>
<td>10</td>
<td>50521</td>
<td>NA</td>
<td>43506</td>
</tr>
<tr>
<td>0</td>
<td>110567</td>
<td>NA</td>
<td>110567</td>
</tr>
</tbody>
</table>

CPU/sec

600
600
600

Table 4: Test time results (#cycles) for p93791.

<table>
<thead>
<tr>
<th>Cmax</th>
<th>tTAM (32bit IO)</th>
<th>dTAM (32bit IO)</th>
<th>tTAM+dTAM (64bit IO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>oo</td>
<td>1720245</td>
<td>1125190</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>1720245</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>1720245</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>1193353</td>
<td>NA</td>
<td>122315</td>
</tr>
<tr>
<td>0</td>
<td>1449193</td>
<td>NA</td>
<td>1449193</td>
</tr>
<tr>
<td></td>
<td>1570566</td>
<td>NA</td>
<td>1570566</td>
</tr>
</tbody>
</table>

CPU/sec

7200
7200
7200

5.5 Heuristic Approach for Wrapper and Transparent TAM Co-Optimization

5.5.1 Simulated Annealing

In the previous section, we observed that the proposed ILP model cannot be solved within a reasonable time for large SoCs. Through the experiments, however, we had the following observations: (1) the number of test sessions that gives the minimum test time is much lower than the number of cores (i.e., serial test schedule), and (2) the ILP model can be solved within a few seconds once the session assignment for each core is done (i.e., sⱼ,k is decided).

Based on the above observations, we present a heuristic method for the wrapper and transparent TAM co-optimization. First, we limit the number of test sessions to a constant value (we used “5” sessions in our experiments). Second, we determine the session assignment for each core outside the ILP model and solve the ILP model with the session assignment. We use the simulated annealing (SA) technique to find the optimum session assignment. The SA algorithm is shown in Fig. 3. The SA algorithm starts to generate an ILP model described in the previous section and solve the ILP model with a randomly generated initial session assignment.

1: Generate an ILP model and an initial session assignment Aᵢ; assign
2: Solve the ILP with Aᵢ and get the test time Cᵢ;
3: Set initial temperature T = Tᵢ;
4: while stop criteria are not met do
5: for i = 1 to Nᵢ do
6: Generate a neighboring assignment Aₜ from Aᵢ;
7: Solve the ILP with Aₜ and get the test time Cₜ;
8: ΔC = Cᵢ - Cₜ (Computation change of cost function);
9: if ΔC ≤ 0 then
10: Set Aᵢ = Aₜ;
11: else
12: Set q = random(0, 1);
13: if q ≤ e⁻ΔC/T then
14: Set Aᵢ = Aₜ;
15: end if
16: end if
17: end for
18: Set new temperature T = β · T;
19: end while

SA algorithm with 5 test sessions. The number in parentheses denotes the relative difference from the original ILP model proposed in Section 3.

Figure 3: Simulated annealing algorithm.
Table 5: Test time results (#cycles) for d695 by SA.

<table>
<thead>
<tr>
<th>Cmax</th>
<th>TAM (14bit I)</th>
<th>dTAM (32bit I)</th>
<th>TAM+dTAM (48bit I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>oo</td>
<td>50383</td>
<td>49615</td>
<td>49894</td>
</tr>
<tr>
<td>100</td>
<td>50383</td>
<td>49615</td>
<td>49894</td>
</tr>
<tr>
<td>25</td>
<td>50383</td>
<td>49615</td>
<td>49894</td>
</tr>
<tr>
<td>10</td>
<td>50383</td>
<td>49615</td>
<td>50383</td>
</tr>
<tr>
<td>0</td>
<td>110567</td>
<td>110567</td>
<td>110567</td>
</tr>
</tbody>
</table>

CPU(sec) 600 600 60 60 60 60 60

Table 6: Test time results (#cycles) for p93791 by SA.

<table>
<thead>
<tr>
<th>Cmax</th>
<th>TAM (32bit I)</th>
<th>dTAM (32bit I)</th>
<th>TAM+dTAM (64bit I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>oo</td>
<td>1720245</td>
<td>1097871</td>
<td>967409</td>
</tr>
<tr>
<td>100</td>
<td>1720245</td>
<td>1097871</td>
<td>967409</td>
</tr>
<tr>
<td>20</td>
<td>1193353</td>
<td>1018728</td>
<td>NA</td>
</tr>
<tr>
<td>10</td>
<td>1449193</td>
<td>1025573</td>
<td>NA</td>
</tr>
<tr>
<td>0</td>
<td>1570566</td>
<td>1246249</td>
<td>NA</td>
</tr>
</tbody>
</table>

CPU/sec 7200 7200 1800 7200 7200 1800

Table 7: Comparison of test time (#cycles) between the previous transparent TAM and proposed transparent TAM when Cmax = ∞.

<table>
<thead>
<tr>
<th>Soc</th>
<th>TAM (serial)</th>
<th>TAM (co-opt)</th>
<th>red(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>d695</td>
<td>620466</td>
<td>620325</td>
<td>-0.6</td>
</tr>
<tr>
<td>p93791</td>
<td>1223481</td>
<td>967409</td>
<td>-20.9</td>
</tr>
</tbody>
</table>

5. However, we cannot get the optimal solution for all the cases. In contrast with the results for d695, the proposed SA based approach can further reduce test time compared to “5TS+ILP” with 4 times shorter computational time. This is because the intermediate solutions provided by “5TS+ILP” for d695 are very close to the optimal solutions while those for p93791 are still far from the optimal solutions. For the cases such as p93791, the proposed SA based approach can explore the solution space effectively within the limited time compared to the original ILP model proposed in the previous section.

Finally, we compare the proposed wrapper and transparent TAM co-optimization method and the previous transparent TAM methods in Table 7. As we explained in Section 2, the previous transparent TAM methods do not consider the wrapper and TAM co-optimization including test scheduling problem to minimize test time. Therefore, only the serial test schedule is possible. Then, we generated the serial test schedule without considering the total increase of interconnects (i.e., Cmax = ∞). In the serial test schedule, we assumed that each core has the maximum wrapper width (i.e., the width of transparent TAM I/O shown in Column 4 of Table 1) for each SoC to minimize the test time. The results for the serial test schedules are shown in Column 2 “TAM serial” of Table 7. Column 3 “TAM co-opt” denotes the test time of the proposed method where we chose the best test time among the proposed three approaches (i.e., ILP using complete sessions, ILP using 5 sessions and SA using 5 sessions) when Cmax = ∞. The proposed method can achieve up to 25% reduction in test time.

6 Conclusion

We have proposed an ILP formulation and SA based heuristic approach for wrapper and transparent TAM co-optimization. To the best of our knowledge, the wrapper and transparent TAM co-optimization including test scheduling problem has been discussed for the first time in this paper. We have made experiments on three SoCs where we showed that the proposed ILP model is effective for small SoCs while the SA based heuristic approach can explore the solution space effectively for large SoCs. The experiments have also shown the effectiveness of the proposed method compared to the previous transparent TAM approaches and the conventional dedicated TAM approaches.

References