Layout-Driven SOC Test Architecture Design for Test Time and Wire Length Minimization

Sandeep Kumar Goel Erik Jan Marinissen

Philips Research Laboratories
IC Design – Digital Design & Test
Prof. Holstlaan 4, M/S WAY-41
5656 AA Eindhoven, The Netherlands

{SandeepKumar.Goel, Erik.Jan.Marinissen}@philips.com

Abstract

This paper extends existing SOC test architecture design approaches that minimize required tester vector memory depth and test application time, with the capability to minimize the wire length required by the test architecture. We present a simple, yet effective wire length cost model for test architectures together with a new test architecture design algorithm that minimizes both test time and wire length. The user specifies the relative weight of the costs of test time versus wire length. In an integrated fashion, the algorithm partitions the total available TAM width over individual TAMs, assigns the modules to these TAMs, and orders the modules within one TAM such that the total cost is minimized. Experimental results on five benchmark SOCs show that we can obtain savings of up to 86% in wiring costs at the expense of <4% in test time.

1 Introduction

Modular testing is becoming increasingly popular for large SOCs. In order to enable modular test development, an embedded module should be isolated from its surrounding circuitry and electrical test access should be provided. Zorian et al. [1] introduced a generic conceptual test architecture that enables modular testing of SOCs. It consists of three elements per module-under-test: (1) a test pattern source and sink, (2) a test access mechanism (TAM), and (3) a wrapper. The TAM provides electrical access to the module-under-test. The wrapper provides the isolation facility, by implementing switching functionality between functional access to the module, and test access (both inward- and outward-facing) through a TAM.

Various wrapper/TAM architectures [2, 3, 4] and automated design procedures for these architectures [5, 6, 7, 8, 9, 10, 11, 12, 13] have been proposed. These design procedures determine the number of distinct TAMs, their widths, the assignment of modules to TAMs, and the wrapper design per module. Most of these design procedures use the SOC *test time* as minimization criterion. This is motivated by the fact that a large SOC typically has a large test data set, which requires a large test application time on an ATE with deep, and hence expensive, vector memories. However, the wiring of TAMs is another important cost factor, which, until now, has received little attention in the publications on automated TAM design. Nowadays, many SOCs implement quite wide, dedicated TAMs, and especially for such SOCs, it pays off to minimize the TAM wire length while designing the TAM architecture. Short TAM wires reduce the required area cost, performance impact, power dissipation, and cross coupling. Optimizing the TAM wire length requires to take the layout positions of the various modules into account.

This paper presents an SOC wire length cost model for TAMs. Subsequently, we formulate the layout-driven test architecture design problem, in which we assume that the layout positions of all modules of the SOC are given. Next we present a new test architecture design algorithm that combines two costs i.e., total test time and total wire length into one cost function and depending on the weight associated with each cost, computes an optimized test architecture. The presented algorithm minimizes the total wire length by assigning neighboring modules as much as possible into the same TAM and by determining a layout-driven optimal ordering of modules connected to the TAM. We show that the problem of determining an optimal ordering of modules connected to a TAM is equivalent to the well-known Traveling Salesman Problem (TSP) [14] and present a greedy algorithm to solve it.

The sequel of this paper is organized as follows. Section 2 reviews the prior work in this domain. Section 3 describes a wire length cost model. In Section 4, we formulate the layout-driven test architecture design problem and present a heuristic algorithm to solve it. Section 5 contains experimental results. Finally, Section 6 concludes this paper.

2 Prior Work

Most SOC test architecture optimization algorithms published so far have concentrated only on test time minimization of the Test Bus Architecture [3]. Chakrabarty [5] described a Test Bus Architecture design approach that minimizes test time through Integer Linear Programming (ILP). Ebadi and Ivanov [6] replaced ILP by a genetic algorithm. In [7], Huang et al. mapped Test Bus Architecture design to the well-known problem of two-dimensional bin packing and used a Best Fit Decreasing algorithm to solve it. Iyengar et al. [8] were the first to formulate the problem of designing TAMs and wrappers in conjunction, in order to minimize test time. Despite its $\mathcal{NP}$-hard character, they solved the problem using ILP and exhaustive enumeration. In [9], the same authors presented efficient heuristics for the same problem. In [10], Iyengar et al. describe a heuristic algorithm for co-optimization of wrappers and TAMs based on rectangle packing for an architecture in which each module connects to a subset of the wires of one common test bus. Later, this approach was extended with precedence, preemption, and power constraints [15].

A major drawback of the Test Bus Architecture is that the wrappers connected to the same test bus cannot be accessed simultaneously,

1530-1591/03 $17.00 © 2003 IEEE

which complicates testing of the circuitry and wiring in between the modules. The TestRail Architecture [4] does not have this drawback. In [11], Goel and Marinissen described two heuristic algorithms for co-optimization of wrappers and TAMs for the TestRail Architecture. The algorithms in [11] work both for modules with fixed-length and flexible-length scan chains, but have as limitation that the total TAM width should be greater than or equal to the total number of modules in the SOC. The same authors removed this limitation and presented improved results for a new algorithm called TR-ARCHITECT in [12].

Most papers that address the issue of minimizing the wire length of the on-chip test infrastructure focus on (module-level) scan chain design. Early papers in this field [16, 17, 18, 19] describe the ordering of the flip flops in a single scan chain. This problem is equal to the well-known $\mathcal{NP}$-hard TSP [14], for which many heuristic algorithms are available. In many practical cases, multiple scan chains are designed, in order to make better use of the available bandwidth over the IC pins to bring test patterns in and out of the circuit, and hence reduce the test application time. This leads to a Multi Traveling Salesmen Problem (MTSP), in which the scan flip flops need to be both partitioned and ordered. Barbagallo et al. [20] describe a genetic algorithm to address this problem; their compute time complexity seems to be an issue, as they report several hours compute time for modules with less than 2,000 flip flops. Marinissen and Tap [21] developed an efficient heuristic for assigning scan flip flops to scan chains, based on the layout positions of the scan flip flops and the scan pins.

The only researchers, to the best of our knowledge, that have addressed the SOC-level issue of TAM design while taking both test time as well as TAM wire length into account are Chakrabarty and Iyengar [13, 22]. [13] extends Chakrabarty's original TAM design approach that minimizes test time through ILP [5] with a rudimentary form of TAM wire length minimization. For every pair of modules, the user's preference for assigning the modules to the same TAM is expressed by a 0-1 constant. The user also specifies how many of these preferences should at least be rewarded in any solution generated by the ILP solver. [22] extends this approach by adding another 0-1 constant for every pair of modules, expressing the user's preference for *not* assigning the modules to the same TAM. While we acknowledge that these papers were the first to add wire length optimization to TAM design, their proposed solution approaches have many shortcomings. They are based on the Test Bus Architecture, which does not allow for module-external testing [11]. The approaches do not support optimization of TAMs and wrappers in conjunction [8]. They work with a fixed user-specified number of TAMs, whereas most users only want to specify the total number of TAM wires, and leave the number of TAMs to the optimization algorithm [8]. Both approaches lack a real wire length model. The binary 0-1 constants only provide a very coarse way to express preferences and do not allow for gradation of layout distances between modules. Despite its $\mathcal{NP}$-hard character, the problem is addressed by ILP, and hence only small problem instances can be handled within practical compute time bounds [9]. And finally: the test time penalty of taking the wire length preferences into account is rather high; up to 65% for the small examples described in [13].

3 Wire-Length Cost Model

Our test architecture optimization is handled by an iterative algorithm. This algorithm needs to evaluate the wire length costs of many (partial) test architectures during the course of its execution. The most accurate wire length costs can be obtained by creating an actual layout of the SOC and the proposed test architecture. However, layout generation is a compute-intensive task, and certainly not one we can afford in our iterative algorithm, in which many times wire length costs need to be evaluated. Therefore, we have developed a wire length cost model, which, on one hand, allows for fast computation, but, one the other hand, is still sufficiently accurate to force our test architecture optimization algorithm to put neighboring modules as much as possible into the same TAM.

Our wire length cost model makes the following assumptions.

1. *Coordinate system*

   - We use only the first quadrant of an orthogonal coordinate system. This assumption is without loss of generality; any SOC layout can be made to meet this assumption with a simple translation. The fact that the SOC is in the first quadrant means that all coordinates will be non-negative numbers, which simplifies our calculations.
   - All coordinates are (non-negative) *integers*. The unit of the coordinates is not specified, but should be consistent for all coordinates belonging to the same SOC. This assumption is again without loss of generality, and meant to simplify our calculations.

2. *SOC layout position*

   - The SOC layout is assumed to be a rectangle, of which the bottom-left corner coincides with the origin $(0, 0)$ of our coordinate system. For rectangular SOCs, this assumption is without loss of generality, as it can be met by applying a simple translation and/or rotation.
   - The coordinates of the center of the SOC layout are specified as (X, Y). From this, and from the fact that the bottom-left corner is at $(0, 0)$, we can calculate the positions of the horizontal boundaries of the SOC layout to be at $y = 0$ and $y = 2Y$, while the vertical boundaries of the SOC are at $x = 0$ and $x = 2X$.

3. *Module layout position*

   - The position of each Module m is specified by a pair of coordinates (x_m, y_m), corresponding to the center of the bounding box of the layout block of m. All TAM wires to and from m are assumed to start and end in (x_m, y_m).

4. *Layout distances*

   - For calculating distances between modules, we use the Manhattan distance instead of the Euclidean distance. This is motivated by the fact that SOC routing channels only allow horizontal and vertical wiring. In addition, the Manhattan distance function simplifies our distance calculations. The distance between Modules m_1 and m_2 is $d(m_1, m_2) = |x_{\mathrm{m}_1} - x_{\mathrm{m}_2}| + |y_{\mathrm{m}_1} - y_{\mathrm{m}_2}|$.
   - For calculating the distance between a module and the SOC boundary, we use the shortest distance between that module and any of the SOC boundaries. Hence, $d(m) = \min(x_m, y_m, 2X - x_m, 2Y - y_m)$. This assumption is based on the idea that the set of pins on which the TAM wires will be multiplexed is not pre-determined.

Note that the assumptions above are compatible with the format of the layout information of the *ITC'02 SOC Test Benchmarks* [23].

In this paper, a TAM r is fully represented by its width $w(r)$ and an ordered list of modules $< m_1, m_2, \ldots, m_{|r|} >$. The wire length for a TAM r is the sum of the distance between the SOC boundary and Module m_1, the distances between all subsequent pairs of modules in r, and the distance between Module $m_{|r|}$ and the SOC boundary, multiplied by the number of wires $w(r)$. The total wire length $l(r)$ of TAM r can be written as follows.

$$l(r) = w(r) \cdot \Big(d(m_1) + \sum_{i=1}^{|r|-1} d(m_i, m_{i+1}) + d(m_{|r|}) \Big).$$

The total wire length for an SOC with a set R of TAMs is the sum of the wire lengths of the individual TAMs in R, i.e., $L(R) = \sum_{r \in R} l(r)$.

A simple example SOC layout to illustrate our wire length cost model is shown in Figure 1. The SOC contains four modules, named A, B, C, and D. The SOC has two TAMs; TAM r_1 of width $w(r_1)$ goes through Modules A and D, while TAM r_2 of width $w(r_2)$ connects Modules B and C.

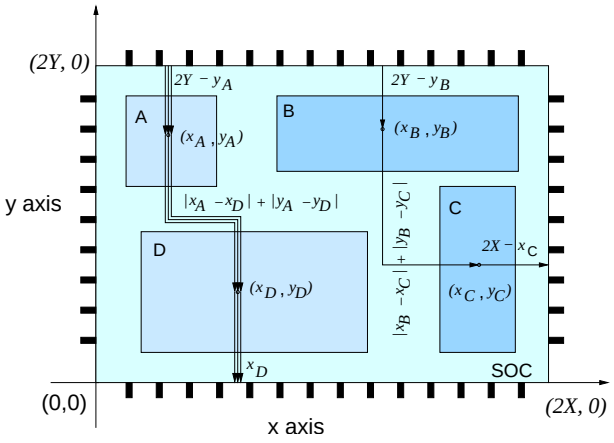


Figure 1: Example SOC layout to illustrate our wire length cost model.

4 Layout-Driven Architecture Design

In layout-driven test architecture design, we want to minimize both the time necessary to complete all SOC tests, as well as the wire length necessary for the TAMs. These two minimization criteria can be conflicting. Therefore, we require from the user to specify the relative weight α of these two cost terms. The problem of layout-driven test architecture design can be formally defined as follows.

Problem 1 [Layout-Driven Test Architecture Design]
Given an SOC with center layout coordinates (X, Y) and a set of modules M. Given for each Module $m \in M$ its center layout coordinates (x_m, y_m), the number of test patterns p_m, the number of functional input terminals i_m, the number of functional output terminals o_m, the number of bidirectional terminals b_m, the number of scan chains s_m, and for each scan chain k, the length of the scan chain in flip flops $l_{m,k}$. Furthermore, given the maximum number of SOC-level TAM wires $w_{\max}$ and the relative cost function weight α ($0 \leq \alpha \leq 1$).
Determine a set of TAMs R (i.e., the TAM widths and ordered lists of modules assigned to those TAMs), and a corresponding wrapper design per module, that satisfy the following conditions: (1) $M = \bigcup_{r \in R} r$ and $\forall_{r_1, r_2 \in R}\ (r_1 \cap r_2 = \emptyset)$, i.e., every module is assigned to exactly one TAM, (2) $\sum_{r \in R} w(r) \leq w_{\max}$, i.e., the summed widths of the TAMs do not exceed $w_{\max}$, such that (3) the cost function $C = \alpha \cdot T(R) + (1 - \alpha) \cdot L(R)$ is minimum (where $T(R) = \max_{r \in R} t(r)$ represents the total test time). □

TR-ARCHITECT [12] proved very effective in minimizing the total test time of SOCs. The original version of TR-ARCHITECT was unaware of the layout-positions of the SOC and its modules, and consequently did not minimize the TAM wire length nor order the modules per TAM. In this paper, we propose a new, layout-driven version of TR-ARCHITECT to minimizes the weighted sum of test time and wire length. Our solution approach consists of two items.

- A procedure WIRELENGTH(r) that optimizes the ordering of modules in r with respect to wire length and calculates the corresponding wire length $l(r)$.
- Adaptations of TR-ARCHITECT, such that when determining the number of TAMs, their widths, and the assignment of modules to TAMs, we avoid inclusion of modules which are far apart from each other in the same TAM.

Both items are described in more detail in the subsequent sections. Note that the procedure WIRELENGTH(r) is used as a subroutine in the layout-driven version of TR-ARCHITECT. However, it can also be used to optimize the wire length of any other TAM architecture, e.g., one generated by the original, non-layout-driven version of TR-ARCHITECT.

4.1 Optimized Ordering of Modules per TAM

The problem of determining an optimal ordering of modules that are connected to a TAM, can be formulated as follows.

Problem 2 [OptimalOrderDesign]
Given are the set of modules $\{m_1, m_2, ..., m_{|r|}\}$ connected to TAM r and its width $w(r)$. For each Module m, the center layout coordinates (x_m, y_m) are given. Also given are the center layout coordinates (X, Y) of the SOC for which the TAM is designed. Determine an optimal order of Modules $m_1, m2, ..., m_{|r|}$ such that the total wire length $l(r)$ required to connect $w(r)$ TAM wires to the modules and the SOC pins is minimized. □

We approach this problem with a two-step solution. First, we determine an ordering of the modules, such that their TAM interconnect wire length is minimized. In the second step, we calculate the overall TAM wire length by adding the wires from the head and tail of the TAM to the SOC pins.

We look at Step 1 as to the problem of finding the shortest path through all nodes in a complete undirected weighted graph. Let $G = (V, E)$ be a complete undirected weighted graph, where vertex $v_i \in V$ corresponds to Module m_i and weight $W(e_{ij})$ on edge $e_{ij} \in E$ represents the wire length cost of connecting Modules m_i and m_j, i.e., $w(r) \cdot d(m_i, m_j)$. This problem is very similar to the well-known *Traveling Salesman Problem* TSP [14], in which a traveling salesman has to find the shortest tour through a given set of cities with pre-defined distances. Our problem is to find the shortest *path* through all nodes, while TSP is after the shortest *tour* through all cities. Our problem can be transformed

into an instance of TSP by adding one node with equal distances to all other nodes. TSP is known to be $\mathcal{NP}$-hard [14]. In practical terms, this means that the time needed to compute an optimal solution increases exponentially with the problem instance size. In order to avoid intractable compute times, we employ an efficient, yet effective heuristic algorithm. Several efficient heuristic algorithms have been proposed in literature to solve TSP. We use a simple greedy heuristic [17] for Step 1. Algorithm 1 lists the pseudo-code of our algorithm.

Algorithm 1 [WIRELENGTH$(r, w(r))$]

```
1   /* Step 1: Ordering of modules */
2   create a complete graph G = (V, E) from modules in r;
3   for all edges e_ij ∈ E {
4       W(e_ij) := w(r) · d(m_i, m_j) };
5   sum := 0;
6   while |V| > 1 {
7       find edge e*_ij for which W(e*_ij) = min_{e∈E} W(e);
8       merge vertices v_i and v_j to form a super-vertex v_{i,j};
9       sum := sum + W(e*_ij);
10      delete edges incident to vertices v_i and v_j;
11      V := V \ {v_i, v_j} ∪ v_{i,j};
12      assign new weights on edges connecting v_{i,j} to other vertices;}
13  /* Step 2: Calculation of total wire length */
14  l(r) := sum + w(r) · (d(m_{v_1}) + d(m_{v_|r|}))
```

Algorithm 1 consists of Step 1 (Lines 1–12) and Step 2 (Lines 13–14). Step 1 consists of an initialization, followed by an iterative loop. In the initialization (Lines 2–5), graph $G = (V, E)$ is built, and the summed wire length so far is set to zero. In the main loop (Lines 6–12), in a greedy fashion every time two vertices are merged into a super vertex. The merge order corresponds to the ordering of modules in the TAM. Once a super-vertex is formed, the two vertices and the edges incident to them are deleted from the set V and E respectively. The weights on the edges connecting the super-vertex to other vertices are re-calculated and re-assigned. This step continues until a single vertex is left. A super-vertex represents a chain of vertices and only the ending vertices of the chain sequences can be connected to other vertices. For example, to connect one super-vertex to another vertex, there are two possible connections. We consider the minimum cost of all possible connections as the weight on the edge connecting the two vertices.

4.2 Architecture Design Algorithm

The new layout-driven version of TR-ARCHITECT uses the same four steps as in the original version [12]: (1) CREATESTARTSOLUTION, (2) OPTIMIZEBOTTOMUP, (3) OPTIMIZETOPDOWN, and (4) RESHUFFLE. However, instead of optimizing the test time in each step, the new version optimizes the total cost C. The details of all four steps are given below.

In the step CREATESTARTSOLUTION, an initial test architecture is created which is then further optimized by the steps to follow. In this step, modules are assigned to one-bit wide TAMs. If there are more TAM wires than modules, each module gets assigned. Otherwise, only the largest $w_{\max}$ modules get assigned. 'Large' is here defined by the test data volume for each module. If we have some modules left unassigned, modules are added iteratively to the TAMs which results in minimum overall cost. Similarly, if some TAM wires are left, these wires are added to the TAMs provided there is no increase in the overall cost C. This step returns a number of unused wires and a set of TAMs R.

The next two steps, i.e., OPTIMIZEBOTTOMUP and OPTIMIZETOPDOWN, try to merge the modules of two TAMs into one new TAM, such that the wires that are freed up in the process can be utilized for an overall cost reduction. The freed up wires can reduce the overall cost in two ways: (1) freed up wires can be used by the bottleneck TAM to minimize T, or (2) L is reduced as less wires need to be routed.

In the step OPTIMIZEBOTTOMUP, the TAM with the shortest test time is merged with another TAM such that there is no increase in C and C is minimum. The procedure assigns the maximum of the widths of the merged TAMs to the newly formed TAM. In this way, the minimum width of the merged TAMs is free for reduction of C. This procedure ends if all TAMs have been merged into one single TAM, or when no further decrease of C can be obtained. Similarly, in step OPTIMIZETOPDOWN, the TAM with the largest test time is merged with another TAM. This step might reduce T as the width assigned to the merged TAM is the summed widths of the merging candidate TAMs.

In the step RESHUFFLE, an individual module is moved from the bottleneck TAM to another TAM, if and only if this decreases the overall cost. It is important to note here that, due to weight dependency on the wire length and the test time in the overall cost function, it is possible that for small values of α, a large number of TAM wires remain unassigned. This is due to the fact that adding one more wire to a TAM directly increases the wire length but does not necessarily decreases the test time. However, it is also possible that if all the wires are added iteratively to the TAM with the largest test time, then the decrease in the overall test time might be larger than the increase in the total wire length. To account for this phenomenon, the proposed TR-ARCHITECT tries to assign all unused wires remained after every step to the TAMs to minimize the overall cost.

5 Experimental Results

In this section, we present experimental results for the proposed layout-driven version of TR-ARCHITECT. For experiments, we used the *ITC'02 SOC Test Benchmarks* [23]. As the original benchmark SOCs do not have any data related to the positions of modules in the SOC layout, we used randomly-generated, but feasible floor plans for the benchmark SOCs. The proposed layout-driven TR-ARCHITECT is able to generate both Test Bus and TestRail Architectures like the original algorithm in [12]. Here, we present Test Bus Architecture results only, but similar results were obtained for TestRail Architectures.

We compare test time and wire length results for three approach. The baseline approach is obtained by the original version of TR-ARCHITECT, followed by a lexicographical ordering of the modules per TAM. The lexicographical ordering represents a random ordering with respect to the layout positions of the modules. The second approach uses the test architecture design obtained by the original version of TR-ARCHITECT, followed by a layout-driven ordering of modules per TAM (as described in Section 4.1). The third, and best approach is based on the new, layout-driven version of TR-ARCHITECT, and includes both layout-driven test architecture design and layout-driven module ordering.

The table in Figure 2(a) presents results for a range of $w_{\max}$ values for five of the twelve benchmarks SOCs [23]. These five SOCs were selected, as they are the only ones for which test time contin-

ues to decrease for increasing values of $w_{\max}$ upto 64 [12]. Columns 3 and 4, present the test time T and wire length L for the original TR-ARCHITECT with random ordering of modules per TAM respectively. Column 5 presents the wire length results for the original TR-ARCHITECT with layout-driven ordering of modules per TAM. Test times are not listed for this approach, as they are the same as the test times in Column 3.

Columns 6, 7, and 8 present the results for the new layout-driven TR-ARCHITECT. For the new layout-driven approach, we tried all values of α from 0.1 upto 0.9 with increments of 0.1. The sixth column of the table lists the values of α for which the new layout-driven TR-ARCHITECT yields the best test time results. For most cases, the minimum test time was achieved for $\alpha = 0.9$. This was to be expected, as this value of α most strongly emphasizes test time. The percentages shown in Columns 5, 7, and 8 express the savings in test time resp. wire length compared to the baseline approach.

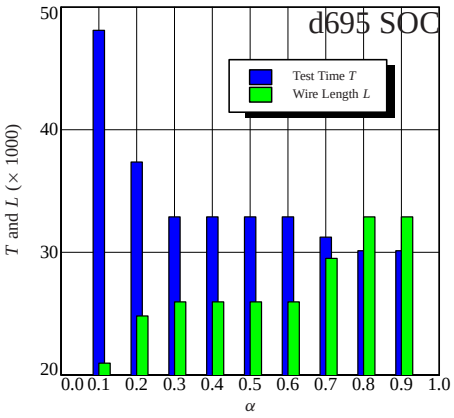
For SOC d695, the savings in L vary from 16 to 65%. It is interesting to note here, that for $w_{\max} = 16$, the new layout-driven approach also improves the test time with 4%. For SOC p22810, the saving in wire length is 58 to 81%, at the cost of an increase of between -3 and 10% in test time. Similarly for SOCs p34392 and p93791, the savings in the wire length vary from 50 to 80% and 48 to 86% respectively, at a penalty in test time in the range of 2 to 6%. For SOC a586710, the savings in wire length are also comparable to the other SOCs, and are in the range of 21 to 61%.

Figures 2(b) and 2(c) show the variation of test time T and wire length L with α for SOCs d695 and p22810 with TAM widths $w_{\max} = 24$ and $w_{\max} = 48$ respectively. The figures confirm that α indeed works as a weighting factor between the two, often conflicting, cost factors.

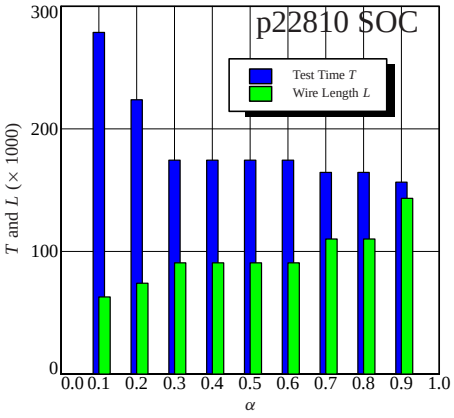
Figure 3 shows an example of the module assignment to TAMs for SOC p93791 with $w_{\max} = 32$ as obtained by the original TR-ARCHITECT (Figure 3(a)) versus the one obtained by our new layout-driven TR-ARCHITECT with $\alpha = 0.7$ (Figure 3(b)). Modules with the same color and pattern are connected to the same TAM. The original version of TR-ARCHITECT divides the total TAM width over three TAMs as shown in Figure 3(a). In this case, modules placed in different corners of the layout are assigned to the same TAM; for example Modules 6, 20, 14, 29 32 and 5, are being put into one TAM. The same is true for other TAMs also. This results in a large wire length for each TAM. The layout-driven version of TR-ARCHITECT generates a TAM architecture as shown in Figure 3(b). The total TAM width is partitioned over nine TAMs. It is clear from Figure 3(b), that only the modules that are physically close to each other are connected to the same TAM, which minimizes the total wire length.

SOC	$w_{\max}$	Original TR-Architect [12]		Layout-Driven Module Ordering		Layout-Driven TR-Architect with Module Ordering				
		T	L	L		α	T		L	
d695	16	44307	86451	63339	-26.7%	0.9	42548	-4.0%	49404	-42.9%
	24	28576	78908	69344	-12.1%	0.9	30132	5.4%	32892	-58.3%
	32	21518	87860	80430	-8.5%	0.9	22438	4.3%	39241	-55.3%
	40	17677	146120	137044	-6.2%	0.9	17677	0.0%	57132	-60.9%
	48	14608	163444	134697	-17.6%	0.9	16194	10.9%	56492	-65.4%
	56	12462	136828	136828	0.0%	0.9	13354	7.2%	66024	-51.7%
	64	11033	87837	87837	0.0%	0.9	11274	2.25%	73736	-16.1%
p22810	16	458068	541750	338608	-37.5%	0.8	462030	0.9%	104189	-80.8%
	24	299718	388607	331032	-14.8%	0.9	298054	-0.6%	147340	-62.1%
	32	222471	288573	234779	-18.6%	0.9	243791	9.6%	120342	-58.3%
	40	190995	415412	355063	-14.5%	0.9	194193	1.7%	94460	-77.3%
	48	160221	358205	310436	-13.3%	0.9	156472	-2.3%	143297	-60.0%
	56	145417	427962	316938	-25.9%	0.9	145417	0.0%	105952	-75.2%
	64	133405	444676	338647	-23.8%	0.9	135571	1.6%	126948	-71.5%
p34392	16	1010821	118596	100316	-15.4%	0.9	1010821	0.0%	59007	-50.2%
	24	680411	263968	211415	-19.9%	0.9	698844	2.7%	137900	-47.8%
	32	551778	259450	219562	-15.4%	0.9	584524	5.9%	78771	-69.6%
	40	544579	474171	362181	-23.6%	0.9	544579	0.0%	93499	-80.3%
	48	544579	474171	362181	-23.6%	0.9	544579	0.0%	93499	-80.3%
	56	544579	474171	362181	-23.6%	0.9	544579	0.0%	93499	-80.3%
	64	544579	474171	362181	-23.6%	0.9	544579	0.0%	93499	-80.3%
p93791	16	1791638	649338	406713	-37.4%	0.9	1791638	0.0%	289826	-55.4%
	24	1185434	513619	376939	-26.6%	0.9	1211149	2.2%	263828	-48.6%
	32	912233	1054240	759708	-27.9%	0.7	946879	3.8%	151676	-85.6%
	40	718005	704481	543507	-22.9%	0.9	745824	3.9%	313788	-55.5%
	48	601450	690570	538308	-22.0%	0.9	617782	2.7%	286215	-58.6%
	56	528925	1125430	904718	-19.6%	0.9	538346	1.8%	291288	-74.1%
	64	455738	764913	662107	-13.4%	0.9	480003	5.3%	181099	-76.3%
a586710	16	41523868	132700	132700	0.0%	0.9	42117536	1.4%	52765	-60.2%
	24	28716501	112157	98547	-12.1%	0.9	28716501	0.0%	82064	-26.8%
	32	22475033	228561	228561	0.0%	0.9	22475033	0.0%	88961	-61.1%
	40	19048835	303370	303370	0.0%	0.8	19048835	0.0%	118018	-61.1%
	48	15212440	411104	377216	-8.2%	0.8	15212440	0.0%	323372	-21.3%
	56	13401034	206520	206520	0.0%	0.9	13401034	0.0%	137550	-33.4%
	64	12510356	232168	232168	0.0%	0.9	12700205	1.5%	182267	-21.5%

(a)

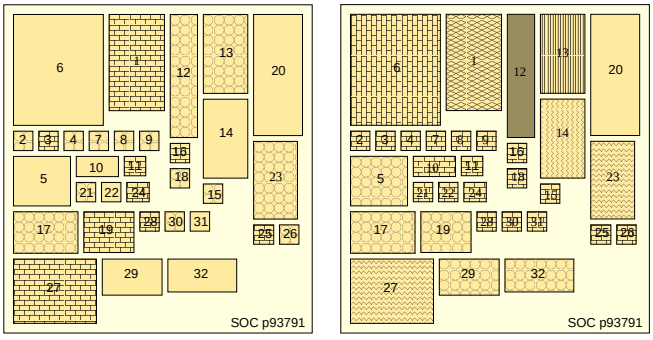


(b)



(c)

Figure 2: Experimental results for (a) test time T and wire length L of original and layout-driven TR-ARCHITECT, and variation of T and L with α for (b) SOCs d695, $w_{\max} = 24$, and (c) p22810, $w_{\max} = 48$.



(a) (b)

Figure 3: TAM module assignments for SOC p93791 with $w_{\max} = 56$, for (a) the conventional TR-ARCHITECT, and (b) the new layout-driven TR-ARCHITECT with $\alpha = 0.9$.

6 Conclusion

In this paper, we have extended an existing SOC test architecture design approach that minimizes test time with the capability to minimize the wire length required for the TAMs. We presented a wire length cost model for TAM design, in which we assume that the layout positions of all modules in the SOC layout are given. Subsequently, we showed that the minimization of test time and TAM wire length for an SOC should be done in conjunction and formulated the layout-driven test architecture design problem.

To calculate the wire length of a TAM, an ordering of modules connected to the TAM has to be found. We show that the problem of determining an optimal ordering of modules with respect to wire length of the TAM is equivalent to the well-known Traveling Salesman Problem (TSP). We used a simple greedy algorithm to solve this problem. Finally, we presented a layout-driven version of TR-ARCHITECT, that minimizes the total wire length by putting neighboring modules as much as possible into the same TAM. Our new approach is based on a user-defined weight α for the two, often competing, cost factors test time and wire length.

For five benchmark SOCs, we compared results obtained from the original TR-ARCHITECT with both random and layout-driven ordering of modules per TAM, with our new layout-driven version. We showed that for all SOCs and a range of $w_{\max}$ values, the new approach results in savings of up to 86% in TAM wire length at the expense of $< 4\%$ additional test time.

Acknowledgements

We thank our colleagues Bart Vermeulen and Jose Pineda de Gyvez for their useful comments on an earlier draft version of this paper.

References

[1] Yervant Zorian, Erik Jan Marinissen, and Sujit Dey. Testing Embedded-Core Based System Chips. In *Proceedings IEEE International Test Conference (ITC)*, pages 130–143, Washington, DC, October 1998.

[2] Joep Aerts and Erik Jan Marinissen. Scan Chain Design for Test Time Reduction in Core-Based ICs. In *Proceedings IEEE International Test Conference (ITC)*, pages 448–457, Washington, DC, October 1998.

[3] Prab Varma and Sandeep Bhatia. A Structured Test Re-Use Methodology for Core-Based System Chips. In *Proceedings IEEE International Test Conference (ITC)*, pages 294–302, Washington, DC, October 1998.

[4] Erik Jan Marinissen et al. A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores. In *Proceedings IEEE International Test Conference (ITC)*, pages 284–293, Washington, DC, October 1998.

[5] Krishnendu Chakrabarty. Design of System-on-a-Chip Test Access Architectures Using Integer Linear Programming. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 127–134, Montreal, Canada, April 2000.

[6] Zahra sadat Ebadi and Andre Ivanov. Design of an Optimal Test Access Architecture Using a Genetic Algorithm. In *Proceedings IEEE Asian Test Symposium (ATS)*, pages 205–210, Kyoto, Japan, November 2001.

[7] Yu Huang et al. Resource Allocation and Test Scheduling for Concurrent Test of Core-Based SOC Design. In *Proceedings IEEE Asian Test Symposium (ATS)*, pages 265–270, Kyoto, Japan, November 2001.

[8] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. Co-Optimization of Test Wrapper and Test Access Architecture for Embedded Cores. *Journal of Electronic Testing: Theory and Applications*, 18(2):213–230, April 2002.

[9] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. Efficient Wrapper/TAM Co-Optimization for Large SOCs. In *Proceedings Design, Automation, and Test in Europe (DATE)*, pages 491–498, Paris, France, March 2002.

[10] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. On Using Rectangle Packing for SOC Wrapper/TAM Co-Optimization. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 253–258, Monterey, CA, April 2002.

[11] Sandeep Kumar Goel and Erik Jan Marinissen. Cluster-Based Test Architecture Design for System-on-Chip. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 259–264, Monterey, CA, April 2002.

[12] Sandeep Kumar Goel and Erik Jan Marinissen. Effective and Efficient Test Architecture Design for SOCs. In *Proceedings IEEE International Test Conference (ITC)*, pages 529–538, Baltimore, MD, October 2002.

[13] Krishnendu Chakrabarty. Design of System-on-a-Chip Test Access Architectures Under Place-and-Route and Power Constraints. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 432–437, Los Angeles, CA, June 2000.

[14] M.R. Garey and D.S. Johnson. *Computers and Intractability - A guide to the theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, 1979.

[15] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. Integrated Wrapper/TAM Co-Optimization, Constraint-Driven Test Scheduling, and Tester Data Volume Reduction for SOCs. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 685–690, New Orleans, LO, June 2002.

[16] Shane Dowd. Post Placement Scan Chain Re-Ordering. *Electronic Product Design*, 17(10):33–38, October 1996.

[17] Chau-Shen Chen, Kuang-Hui Lin, and TingTing Hwang. Layout Driven Selecting and Chaining of Partial Scan Flip-Flops. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 262–267, Las Vegas, NV, June 1996.

[18] K.-H. Lin, C.-S. Chen, and T.-T. Hwang. Layout-Driven Chaining of Scan Flip-Flops. *IEE Proceedings - Computers and Digital Techniques*, 143(6):421–425, November 1996.

[19] Kazushi Nakamura et al. Scanpath's Wire Length Minimization and Its Short Path Error Correction. *NEC Research & Development Journal*, 38(1):22–27, January 1997.

[20] S. Barbagallo et al. Scan Chain Partitioning and Re-ordering Based on Layout Information: An Industrial Experience. In *Proceedings Design, Automation, and Test in Europe (DATE) – Designers Track*, pages 123–127, Paris, France, March 1998.

[21] Erik Jan Marinissen and Mireille Tap. Layout-Driven Scan Chain Partitioning and Ordering. In *IEEE European Test Workshop (ETW)*, Saltsjobaden, Sweden, May 2001.

[22] Vikram Iyengar and Krishnendu Chakrabarty. Test Bus Sizing for System-on-a-Chip. *IEEE Transactions on Computers*, 51:449–459, May 2002.

[23] Erik Jan Marinissen, Vikram Iyengar, and Krishnendu Chakrabarty. A Set of Benchmarks for Modular Testing of SOCs. In *Proc. IEEE International Test Conference (ITC)*, pages 519–528, Baltimore, MD, October 2002.