Hindawi Publishing Corporation
VLSI Design
Volume 2012, Article ID 312808, 11 pages
doi:10.1155/2012/312808

Research Article

A Graph-Based Approach to Optimal Scan Chain Stitching Using RTL Design Descriptions

Lilia Zaourar,[1] Yann Kieffer,[2] and Chouki Aktouf[3]

[1] *SOC Department, LIP6 Laboratory, University Pierre and Marie Curie, 4 Place Jussieu, 75252 Paris Cedex 05, France*
[2] *LCIS, Grenoble Institute of Technology and University of Grenoble, 50 Rue Barthélémy de Laffémas, 26000 Valence Cedex, France*
[3] *DeFacTo Technologies, 167 Rue de Mayoussard, 38430 Moirans, France*

Correspondence should be addressed to Lilia Zaourar, lilia.zaourar@lip6.fr

Received 30 April 2012; Accepted 6 November 2012

Academic Editor: Shantanu Dutt

The scan chain insertion problem is one of the mandatory logic insertion design tasks. The scanning of designs is a very efficient way of improving their testability. But it does impact size and performance, depending on the stitching ordering of the scan chain. In this paper, we propose a graph-based approach to a stitching algorithm for automatic and optimal scan chain insertion at the RTL. Our method is divided into two main steps. The first one builds graph models for inferring logical proximity information from the design, and then the second one uses classic approximation algorithms for the traveling salesman problem to determine the best scan-stitching ordering. We show how this algorithm allows the decrease of the cost of both scan analysis and implementation, by measuring total wirelength on placed and routed benchmark designs, both academic and industrial.

1. Introduction

The design flow of an integrated circuit (IC), meaning the software applications that allows the designer to move from its specification to its concrete realization, involves many stages of optimization problems, usually from system level to layout [1–3]. Indeed, the realization of an IC is a costly process both in time and money, and requires large engineering resources. In addition to technological advances, which continuously improve the efficiency of chip manufacturing techniques, as the fine prints on silicon for example, the continuous increase of the power of computers offers great opportunities for improving the process of designing and manufacturing complex ICs.

With both recent advances of the semiconductor industry and new market constraints, Time To Market (TTM) and product quality are becoming major issues. The circuit must meet flawlessly customer expectations in terms of functionality, speed, quality, reliability, and cost. In such a challenging economic environment, and given the significant level of complexity which is reached by the IC, manufacturing testing is more than ever an important factor in the design problem. Today, chip testing should be short, efficient, and cost-effective. A significant amount of research work is ongoing with a focus on complex design-for-test problems at both universities and industry. Design For Test (DFT) techniques are becoming a key since they are considered during the chip design process and flow. Cost of manufacturing chips averages the cost of testing them. So the semiconductors community needs low cost and high quality test solutions. New and efficient DFT solutions are greeted with higher expectations than ever. Most current DFT solutions result in unpredictable design development time and development costs and directly impact (TTM).

In this context, DFT tools are receiving growing attention with the advent of core based System On Chip (SOC) design. In particular, when cores from different vendors are integrated together on the system on chip (as it is often done nowadays), the difficulty level of testing grows rapidly. Among the most apparent issues are core access, system diagnosis, test reuse, test compaction, tester qualification and Intellectual Property (IP) protection. The ITRS (International Technical Roadmap for Semiconductor) identifies key technological challenges and needs facing the semiconductor

industry through the end of the next decade. Difficult near-term and long-term testing and test equipment challenges were reported in [4].

Several of the DFT solutions for ASIC and ASIP designs are based on the internal Scan DFT technique. Scan is potentially an efficient technique, if used properly. Currently, full Scan is the most widely used structured DFT approach where all the design sequential elements belong to the scan architecture or scan chains [5]. A Scan-based DFT architecture provides the ability to shift information by scanning the set of states of the circuit. All flip-flops are chained to each other to allow the introduction of test vectors as input and retrieval of test results as output. Thus, the resulting shift register is fully controllable and observable through the primary inputs and outputs of the circuit. This makes Scan testing widely adopted for manufacturing testing and other purposes such as silicon debug. However, we do care about hardware overhead (area, pin count, and so on), performance penalty, and extra design effort that may be associated with full scan insertion during the chip design process. Therefore, the primary objective in DFT is to automate the insertion of the test logic and to minimize the impact of test circuitry on chip performance and cost. Thus it is important and essential to optimize the scan chain insertion process.

However, traditional Scan solutions make such an extension very difficult since engineers need to handle gate-level netlist without taking benefit from what happens during synthesis optimization. Also, Scan implementation decisions are considered post synthesis. This is too late in comparison to RTL design decisions. Adopting Scan at the Register Transfer Level (RTL) will cover new design and manufacturing needs, strengthen verification, and consolidate reusable design methodologies by closing the gap between RTL and design for test. There are many advantages to inserting scan at the RTL level like: benefiting from the synthesis process (i.e., better optimization in terms of area and timing), the ability to debug testability issues early in the design flow, and leveraging the optimization done by the synthesis tool. The possibility to insert scan at the RTL dates back to the late nineties [6–8]. Although this idea did not get a widespread attention in the meantime, an EDA tool that lets one do it, namely HiDFT-SIGNOFF by DeFacTo Technologies, has appeared and is commercially available. Its main use today is to help debug testability issues early on in the design flow, thus helping avoid costly iterations around the synthesis step. To achieve this purpose, Scan chains are introduced at the RTL, bringing to light any problems in doing so—like noncontrollable Flip-Flops (FFs), for example. Then the scanned design can be used in other estimates, but will be discarded as far as the main flow is concerned: the real scan chains that will end up being implemented in the chip are still inserted at gate-level, using a traditional design flow. Before one can insert the actual scan chains at the RTL, one big hurdle has to be overcome. The impact of scan insertion on the design cost and performance can be important if the ordering of the FFs in the scan chain is not picked carefully. But the information allowing for such a choice, namely placement information of the FFs, is only available at the back-end of the design flow, far away from the point where the RTL code for scan must be finalized. Our goal in this work is to develop a tool to automatically generate optimal scan chains at RTL in terms of area and additional test time for a given circuit, while respecting a number of electronics constraints.

To analyze the optimal design location where full scan chains need to be inserted, the following considerations are required. First, to implement a scan chain, one has to have knowledge of the Flip-Flops (FFs) of the design. Second, the best place to insert any new task in a flow is at the earliest. The motivations for an early scan insertion are threefold: the complexity of the objects grow as one goes forward in a flow, so data handling gets more costly; whatever is added to the design is better if integration happens earlier in the flow; and finally, should the treatment lead to iterations—either redesigning or iterations in the flow—the sooner the iterations, the lower the cost. Third, at the point where insertion is finalized, all the information required for the insertion has to be available. If this information depends at least partly on the insertion itself, this leads to iterations—see below. Fourth, insertion can start at some level in the flow, and end at a later point in same. Again, this can be undesirable, but also unavoidable once some design decisions on the insertion process have been taken. Since FFs are usually known after synthesis, once a gatel-level netlist is available, this sounds like a reasonable point in the flow to insert scan. But the replacement of normal FFs by scan FFs has to be done before synthesis ends, since timing closure is affected by it. We will see in the next section that FFs detection can happen before synthesis.

On the other hand, we will argue in Section 3 that the scan stitching ordering is an important part of making scan insertion seamless. To compute an optimized ordering, the best information to have is that of the placement of the FFs in the layout. If one uses placement information, scan insertion should be finalized during placement and routing (at the earliest). This is the classical scheme for scan insertion, and it is this one that is used in industrial EDA tools for scan insertion. This is why all scan insertion tools today either provide a costly ordering, or must decide the ordering in several phases. The next paragraph reviews the state of the art, while keeping an eye on these different criteria.

The traditional way to insert full scan in a design is to replace the FFs by scanned FFs during synthesis; possibly connect them into a chain; and then during place and route iterations, (re)connect them to try to minimize the total Manhattan length of the added connections. The exact place where one should reoptimize the ordering of the chain is a matter of debate, and heavily depends on the flow used. For an in-depth discussion of this topic in the case of the Synopsys flow, see [9].

Also, scan implementation decisions are considered post synthesis. This is too late in comparison to RTL design decisions. Adopting scan at the RTL level will cover new design and manufacturing needs, strengthen verification and consolidate reusable design methodologies by closing the gap between RTL and design for test. There are many advantages to inserting scan at the RTL level like benefiting from the

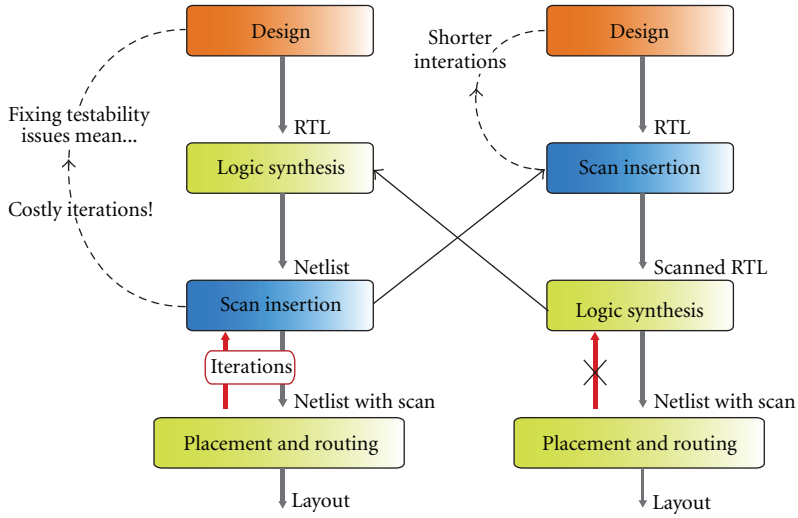


Figure 1: Classical scan insertion versus our method.

synthesis process (i.e., better optimization in terms of area and timing).

Most of the work done on scan chains insertion and stitching ordering optimization assumes that placement information is available [9–14], meaning that it is actually reordering optimization. In all the cases, scan (re)ordering is reduced to the problem of the Traveling Salesman Problem in a suitable graph. An additional assumption is added in [15]. The author proposes a procedure to find and break intersections. This is also done to reduce the overal wire length.

The knowledge of FFs is necessary to implement a scan chain. It has been suggested several times independently that one does not need a netlist to have knowledge of FFs of the design; this is the basis of higher-level scan insertion [6–8, 16, 17]. Although fault models are defined in terms of gates and nets, scan insertion itself can actually be done at the RTL. This is achieved by describing the behaviour of the scan chain directly at the RTL. It will then be translated by the synthesis tool into added nets between FFs, and multiplexers in front of FFs. No scanned FFs are used in this case; we give a more detailed account in the Section 3. It would seem that inserting scan at the RTL makes the problem of scan stitching reoptimization much worse. This may be one reason why in that case, the reoptimization possibility is commonly dropped, in favour of a single optimization at the time of insertion.

Most studies on higher-level scan either do not mention optimization [7], or mention local optimizations only, meaning trying to reuse bits of functional paths for the scan path, thus reducing the need for added nets and additional multiplexing logic [6, 16–18]. In [8] an attempt is made to achieve both local and global optimizations by carefully delineating a graphic model for the scan ordering problem. This graphic model rests on an analysis of the RTL source code. Unfortunately, the algorithm they propose was never implemented [19], and thus could be tested only on small designs.

In this paper, we try to close the gap between RTL Scan insertion and actual scan stitching optimization, while retaining the model of one-time insertion without any later reoptimization see Figure 1.

More specifically, we tackle the following optimization problem: given an RTL description of the design (Verilog or VHDL) we have to find a stitching ordering of the memory elements in the scan chain, which minimizes the impact of test circuitry on chip features (area, power) while keeping testing time at its minimum.

The remaining of this paper is organized as follows. Section 2 presents how our approach to RT-level scan is novel. Section 3 describes the problem of scan chain insertion at the RTL. It explains how RTL scan is implemented and gives arguments as to why wirelength is an efficient global measure of the scan insertion cost. In Section 4, we propose our RTL scan stitching algorithm together with the graph models on which it is based. Implementation, experimental results on both academic and industrial designs are given in Section 5, followed by discussions that RTL scan is a viable alternative to gate-level scan. Finally, Section 6 concludes the paper and points out some research directions for future work.

2. Original Algorithmic Content of This Work

To the best of our knowledge, this work is the first to offer a formal treatment of the scan stitching ordering problem as a discrete optimization problem in the case of RT-level scan insertion. We formalize the problem; give reductions from the several-chains problem to the one-chain problem; and solve the one-chain problem in two steps. The first step, which can be seen as a kind of preprocessing, allows us to build a graph representative of the actual optimization. Then in a second step, algorithms for the TSP are adapted to actually build the chain.

A part of this work has been presented before in [20] and recenlty in [21, 22]. The present paper is aimed at

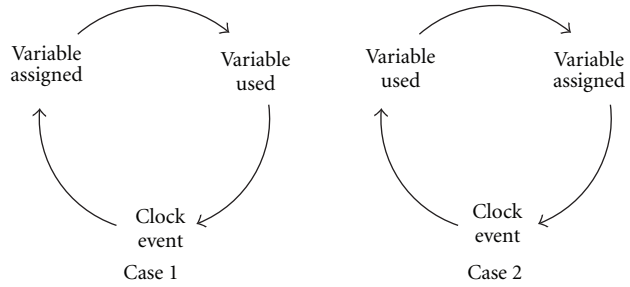


Figure 2: Two use cases for variables in VHDL.

giving a more complete presentation for both theoretical and experimental contributions. In fact, we give here all of the theoretical basis and algorithmic elements of this work including detailed implementation that has not been presented before. Also some numerical results and comparisons resulting are added.

The main contributions of the paper are summarized below:

(i) we analyze what is a suitable objective for measuring the quality of scan stitching orderings,

(ii) we give a mathematical formulation of the problem of scan insertion at the register transfer level (RTL),

(iii) we give two reduction procedures to solve the several-chains variants based on a routine solving the one-chain case,

(iv) we solve the one-chain case in a two-steps approach,

(v) we evaluate our algorithm on both academic and industrial designs.

Our algorithm provides a basis for considering the implementation of scan chains as soon as the RTL of a block is available; the authors think that the lack of optimization has been a big obstacle to its adoption in design flows. The methodology has been validated by our industrial partner DeFacTo Technologies in the tool HiDFT-SIGNOFF, it is a first step towards considering the integration of scan at the RTL.

3. Scan Insertion at the RT-Level

We illustrate Scan insertion at RTL with the VHDL language. Scan insertion is a three steps process: first identify which signals and variables will give rise to flip-flops in the netlist; second, decide which ordering will be used to chain the FFs together; third, change the RTL code to offer a (new) testing mode for the design.

Since the object of this work is the second step of this whole process, we now restrict our attention to its first and third steps. This section is included mainly to make this article self-contained; for a more thorough presentation, the reader is referred to [6–8].

We first present how flip-flops are identified; then we illustrate RTL edition for introducing scan at RTL; finally, we examine what could be a good measure of the impact of scan insertion on a design.

```
process(clock)
begin
   if clock' event and clock='1' then
      Z <= X or Y;
      Y <= X and Y;
   end if;
end process;
```

Algorithm 1: Example VHDL process.

```
process(clock)
begin
   if clock' event and clock='1' then
      if scan_en='1' then
         Y <= X;
         Z <= Y;
      else
         Z <= X or Y;
         Y <= X and Y;
      end if;
   end if;
end process;
```

Algorithm 2: VHDL process enriched with scan code.

3.1. Flip-Flops Identification. FFs identification is realized process by process. Each process is searched for variables and signals. All signals in a process will be translated into a FF in the netlist. To determine which variables will give rise to FFs, one has to first identify clocking signals. Then two cases can happen. To determine which applies in a particular case, one has to recall that processes execute cyclically—see Figure 2.

Either the variable is assigned between the clocking event in the process and the point where the variable is used (case 1). In that case, no FF is generated. Or the clocking event lies between the point where the variable is assigned and that where the variable is used (case 2). In that case, memorization has to occur, and a FF is generated.

3.2. Test-Mode Introduction by RTL Edition. To illustrate the introduction of a scan chain at RTL through RTL code edition, we consider the simple process in Algorithm 1.

In order to introduce a testing mode behavior to the design, we simply describe in VHDL what the process is going to do in test mode. The additional VHDL code is shown in Algorithm 2 in boldface; it consists of a conditional statement on the value of the scan_en signal, and assignments expressing the functional chaining of the registers of the process. Note that this (too simple) example is misleading in showing a doubling of the size of the RTL code; code growth for realistic designs is much less than that.

3.3. Wirelength as the Prime Parameter for Stitching Ordering. In most of the literature about scan ordering optimization, wirelength is the objective used to guide the optimization process. In these works, placement of the FFs is known; so wirelength is a quantity that is directly available during

optimization. We will give our own argument for considering wirelength the most useful parameter to optimize stitching orderings.

When adding logic to a design for non-functional reasons, one tries to minimize the increase in size of the design. Additional area due to scan comes in two parts. First, FFs have to be instrumented either into scanned FFs, or with the help of a multiplexer at their input. In the former case, the area cost depends only on the number of FFs in the design; in the latter, the area cost can be less than the maximum if some optimization takes place during synthesis. In both cases, that cost is bounded independently from the stitching ordering. Second, wires have to be added between the output net of a FF and the next FF in the scan chain—either the scanned FF, or one input of the multiplexer in front of it. This does not represent an area cost in itself, since routing happens in the upper metallic layers. But it does add to the difficulty of placement and routing, with the possibility to have a very degraded situation if the stitching order is not chosen with care. In that case, the area can grow if routing is not possible anymore with the available space. We consider this a much more important factor than the additional cell area; therefore our measure of ordering quality will be based on it.

It is not possible to express the impact of added wires on placement and routing as a simple function of these wires; the impact will depend also on the sparsity of the design. But since we take a worst-case approach to the impact of wires, we will use the added wirelength due to scan insertion as our optimization function.

This choice will be validated by the variability that is observed on this parameter—see Section 5.

We now review quickly all the other parameters scan insertion could have an impact on.

In order to help ensuring timing closure, the maximum of the length of the added wires would be a reasonable measure. But since in our case synthesis happens after scan insertion, it will be up to the synthesis-and later on, placement and routing-tools, to ensure timing closure, using all the flexibility of gate sizing to achieve it. Note also that ensuring a low total added wirelength means that not too many added wires will be long, hence lowering the impact on timing, and the additional load for the synthesis tool.

Power while in functional mode grows with wirelength; so minimizing wirelength will also lower the impact on functional power. Power while in test mode is dominated by switching: the values shifted through the scan chain force the values in FFs to change more often than they would in functional mode, leading to power supply issues. A common remedy to this condition is to take benefit of don't-care values in test vectors to fill them in a way to minimize switching. This also works with RTL-scan. There have been many attempts in other works at minimizing power during test mode by changing the stitching ordering. We have not followed this trail; it would be worthwhile to try combining it with our approach based on minimizing wirelength.

Finally, observability and controllability are not impacted by the stitching ordering. We note here that although combinatorial parts of the circuit have the same observability and controllability in a full scan design implemented both at gate and at RT-level, in the case of RT-level scan, there will be more stuck-at faults, since the multiplexers are now no more combined with FFs. Hence fault coverage values tend to differ, although not much, between both methods for scan insertion.

Having established that wirelength should be minimized, we now turn to the precise description of the optimization problem we will consider in order to try to find low wirelength stitching orderings.

4. Stitching Algorithm for Optimal Scan Chain Insertion at the RT Level

To determine the scan chains that best meets the needs of different integrated circuit designers, while maintaining maximum constraints and restrictions related to the electronic problem, we propose a new scan stitiching algorithm for the automatic insertion of optimal scan chains at the RTL. Our algorithm is structured into two phases, as shown in Figure 3. The first phase aims to build a graph-based model to translate the problem and its constraints from an RTL description to a generic mathematical model, in the form of an undirected graph. In a second phase, scan chains are determined through computations in that graph. The stages in each phase are outlined in the next paragraphs.

4.1. Phase 1: Graph Extraction. The challenge here is to be able to take into consideration still at high level what is going to happen later on during the placement and routing (P&R) steps. To build this model, we extract from the RTL description information on the expected proximity of the memory elements in the layout.

This phase is devised into three steps, as follows.

4.1.1. Design Elaboration (Lightweight Synthesis). First, we perform a preprocessing called Elaboration. Although scan logic can be described at RTL, the very elements of which fault models are talking—nets, gates and FFs—do not exist yet at this level. Hence we need first to translate the RTL code into these elements. This is what synthesis does. But it is not feasible to have a full synthesis at this step in the flow. Once RTL-scan is inserted, the synthesis step still has to be done; we do not want to duplicate that effort.

Our solution relies on a lightweight synthesis as the first step of the RTL scan insertion. This synthesis is done in terms of a virtual library of generic (non-physical) gates; it does not try to optimize logic, timing or gate sizes; it does not need the user to do any fine-tuning; in short, it is done transparently. The user of the scan insertion tool only provides his RTL code, and will get back a scanned RTL code: he will never see the netlist that comes out of the lightweight synthesis. Indeed, in our method, this netlist serves only one purpose, namely graph extraction.

4.1.2. Building the D-Graph. The second step after the design elaboration is to build the undirected D-graph (for Design graph), denoted by $G_D = (V_d, E_d)$. It represents the RTL description of the design. It is constructed as follows (cf.

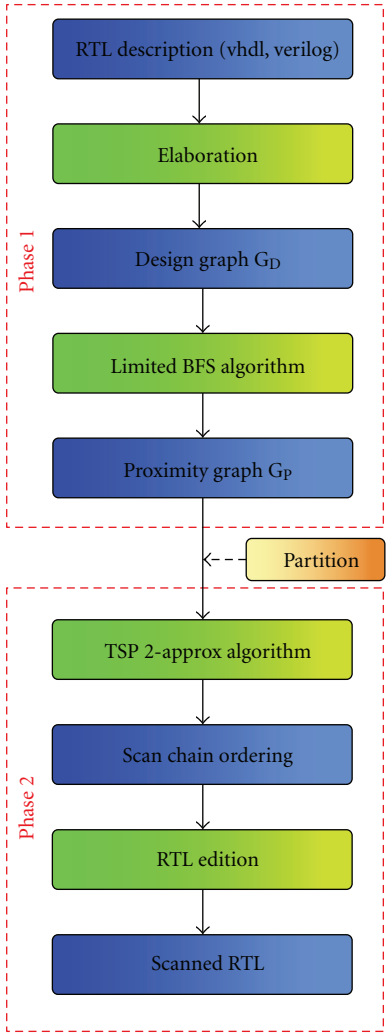


Figure 3: Stitching algorithm for optimal scan chain at RTL.

Figure 4). Each memory element, gate and net from the elaborated netlist corresponds to a vertex (V_D) of the graph G_D. There is an edge (E_D) between two vertices if and only if the corresponding elements in the RTL description or in the netlist are connected.

Note that the vertices are partitioned into two sets: nets on one hand, and gates and memory elements on the other; every edge in the graph has one end on each side. Therefore, G_D is a bipartite graph.

The graph G_D is undirected because it only aims at giving indications on how close memory elements might be located at the placement step; the direction of information flow along the nets is of no consequence for the decision in our algorithm to stitch together two FFs.

4.1.3. Building the P-Graph for Memory Elements Proximity. The third step is to extract information from the design that is necessary for scan chain stitching optimization (second phase). This information will be given in the form of an edge-valuated Proximity Graph (in short *P*-graph), to which we will now refer to as G_P.

The *P*-graph G_P will be built from the graph G_D as follows: the vertices (V_P) are the memory elements of the design; there is an edge (E_P) between two vertices if there is direct electronic connection in the design between the memory elements they represent; this connection is not allowed to go through other FFs. Additionally, the edges are labeled by the length of a shortest non-oriented path between the corresponding vertices in the graph G_D. The weight on edge ij will be denoted w_{ij}. It represents the Path lengths between the pair of memrory elements that it matches.

Path lengths are restricted to a threshold value t which is a fixed parameter of our algorithm: for a longer (shortest) path between two memory elements, no edge is put in G_P.

To compute G_P, we use a variant of the algorithm of Breadth First Search (BFS) [23] that limits the depth of the exploration to find the limited shortest path. This algorithm is called Limited-BFS in the following. The graph computed with the help of Limited-BFS is used to estimate a probable geometric proximity of memory elements in the Layout.

Figure 4 illustrate the construction of G_D and G_P on an example design.

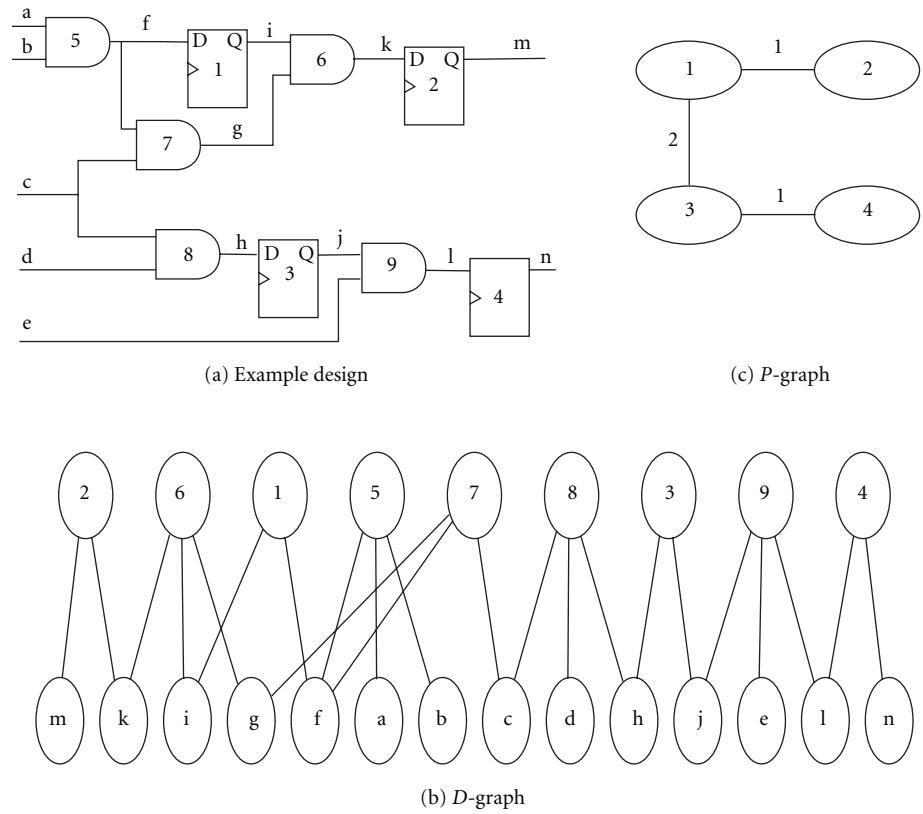
4.2. Phase 2: Construction of Scan Chains. Using this new formalism, the chaining of memory elements in the circuit corresponds to a partition of the vertices of G_P (representing the FFs of the design) into sequences of vertices, each sequence representing one scan chain. An important constraint in scan chaining asks for all chains to have the same number of FFs. Also, the sequences should preferably use edges from the graph G_P, although adding connections is a possibility. The cost of going from v_i to v_j in the sequence is counted as the length of a shortest path from v_i to v_j in the graph G_P; hence it will be w_{ij} if the edge ij is present in G_P.

In the case of a single scan chain, our problem reduces to the Traveling Salesman Problem [23]. Still, to apply TSP algorithms, the whole cost matrix for each pair of vertices of the graphs has to be precomputed; this is not feasible for designs with more than a couple tens of thousands FFs.

Also, single scan chains are not an option for big designs, where testing time would be prohibitive if scanning were not done using more than one chain.

Before explainning the second phase, we present two algorithmic devices to reduce the general chaining problem to the single-chain case. Then we show how our problem can be reduced to the Traveling Salesman Problem and some algorithms to solve it. Finally, we describe the two steps of the second phase.

Chain Splitting. The simplest way of reducing the several-chains scanning problem to its single-chain subcase is by appropriate post-processing. Once one has computed a single scan chain for the whole design, this chain can be split into the desired number of segments (the actual scan chains). This device is really fast, and it is not expected that the quality of the output of the whole process will be much degraged as compared to that of the single-chaining algorithm. Also, it is very easy to implement, and is our recommandation for the cases where fast enough single-chaining algorithms are available.



(a) Example design

(c) *P*-graph

(b) *D*-graph

FIGURE 4: Example design and associated graphs G_P and G_D.

Graph Partitioning. A more elaborate device is the preprocessing of the graph G_P with the help of a graph partitioning algorithm. If s scan chains are called for, the graph G_P should be partitioned into s parts having equal or similar numbers of vertices. Then finding scan chains for the whole design reduces to finding a single chain for each part of G_P.

Even if the partitioning is handled by an appropriate algorithmic package, implementing this solution is not as easy as chain splitting. But using it brings with it another benefit: the TSP problems to solve in this case are restricted to the length of the scan chains. For test application reasons, these are in actuality limited; although technically feasible, one seldom meets scan chains of 10000 FFs.

Hence this solution helps ease up the problem of computing the costs matrix for the input of the TSP, which is actually the longest step of our methodology.

Another benefit is that when using partitioning, the whole method scales with only a linear increase in running times if the maximum size of scan chains is kept constant.

TSP Algorithms. We now discuss how TSP algorithms can be applied for the single-chain stitching problem.

There are two distinct frameworks for applying TSP algorithms to our problem. The first one allows any algorithm to be used, but it puts constraints onto the size of input graphs that can be fed to it.

The second one is the particular case of an algorithm that can be used directly on G_P, without having to compute costs for edges that are not present in G_P.

Using a Generic TSP Algorithm. The input of a TSP routine is a symmetric matrix representing the costs for edges of a complete graph. In order to apply a TSP algorithm to solve the chaining problem, one needs first to compute the whole costs matrix (except diagonal elements). In our model, we attribute (as cost) to a pair ij that is not an edge of G_P the length of a shortest path between i and j in G_P. The post-processing is very simple: just turn the hamiltonian cycle returned by the TSP routine into a hamiltonian path by removing any edge.

It is the preprocessing step that imposes a serious limitation on the possibility to use this scheme. Indeed, the G_P graph obtained even with small values of the threshold parameter for the Limited-BFS step tends to be quite dense; hence any all-pairs shortest path algorithm puts limit to the input size either through space or time complexity, or both. In practice, it may be feasible to handle 5000 FFs or 10000 FFs blocks, but it is difficult to go much higher and keep a reasonable computation time for design scanning.

If only designs smaller than this are to be treated, then this method is definitely worth trying, all the more that one has then the ability to test and compare different TSP algorithms.

4.2.1. Scan Chain Ordering Using the 2-Approximation for the TSP. A famous textbook algorithm for the TSP, and also a typical example of an approximation algorithm with guaranteed quality, the 2-approximation for the TSP also has nice properties when used on the scan chaining problem.

An algorithm for a minimization problem is called an α-approximation when the solution returned by the algorithm is guaranteed to be no more than α times higher than the optimal solution. Usually, such algorithms presented in the literature are polynomial-time algorithms for NP-hard problems: since the requirement to find an optimal solution is dropped, polynomiality can be recovered.

It may seem unintuitive that one can prove a quality bound without even knowing the value of the optimal solution. Actually, such a proof can be derived through the use of a lower bound on the value of the optimal solutions, through showing that the solution given is less than α times this lower bound.

In the case of the TSP, two approximation algorithms are found in every textbook on the subject: one has an approximation factor of 2, (we call it "the 2-approximation for the TSP"), and the other is Christofides' algorithm, with an approximation factor of 1.5. Both use in their proof, as a lower bound on the optimal tour length, the value of a minimum spanning tree of the graph.

Please note that these approximation ratios are only proven theoretical bounds, and are not enough to compare the empirical behavior of these algorithms. Still, Christofides is bound to give solutions that are not more than 50% away from the optimal one; this would entice one to use this algorithm.

Alas, Christofides uses Weighted Perfect Matching in bipartite graphs as a subroutine, which has to impacts on its use. First, weights have to be precomputed, and we stumble against the same blocks as mentioned in the previous section. Second, Weighted Perfect Matching needs $O(n^3)$ operations, seriously limitting the size of its input.

Without any regret, we turn now to the 2-approximation algorithm, which is seldom considered a useful practical choice because of its loose guaranty, and not-so-good performance on common benches.

The algorithm is in two steps. First, a minimum spanning tree is computed; then a root is chosen, and the tree is explored and vertices ouput with postfix ordering.

Only the first step actually looks at the input graph; and its only requirement is that the graph be connected. If G_P is disconnected, we reconnect it by adding arbitrarily edges between connected components until it is connected.

Thus, using the 2-approximation, we bypass the precomputation of the cost matrix. This means that bigger input graphs can be considered if this algorithm is used, as compared to other TSP algorithms.

Although this algorithm was meant for the TSP, that is for an input being a complete graph with values on all edges, one can prove that the 2-approximation guarantee still holds when the algorithm is applied to a connected graph. In this case, the tour cost has to be understood as the sum of the cost of the edges, where edge costs for inexistent edges are length of shortest paths in the input graphs—hence our choice to model the cost of inexistent edges in this way.

4.2.2. RTL Edition. The final step is the edition of the RTL code of the design. In this step, we insert in the original RTL code/description the additional RTL constructions required to implement the scan testing logic.

5. Implementation and Experimental Results

5.1. Implementation. The method we propose to optimize the stitching ordering of RTL scan chains has been implemented in an experimental version of HiDFT-SIGNOFF. This tool relies on a commercial software library for the parsing and elaboration of the design. This elaboration step is what we called "lightweight synthesis" in Section 4.1.1. The generic library of gates used for the elaboration is the one that comes with the software library.

HiDFT-SIGNOFF has the ability to stitch FFs into several scan chains. This possibility is important from the practical point of view: making several scan chains is the easiest way to reduce test application time. HiDFT-SIGNOFF implements the two strategies presented in Section 4 for handling several chains. One is to stitch a unique scan chain, and then split it into even parts. The other is to first partition the design, and then to stitch a scan chain in each part.

For graph partitioning, the METIS library [24] is used. METIS is one of the most widely used libraries for graph partitioning. It includes two strategies P_{MET} and K_{MET}. P_{MET} uses a multilevel recursive algorithm. The second one, K_{MET}, is implemented with an algorithm of K-partition such that the number of vertices per partition is equal. We use here the second one to partition the graph G_p into K parts.

5.2. Experimental Setup. In order to validate our method, we did experiments on a number of designs, both in VHDL and Verilog. We used the benchmarks 99 ITC [25] which are reference designs for testing integrated circuits. Indeed, these designs have been used extensively in the literature for integrated circuits testing. For each design, a complete description of the circuit with its value is presented on the website. We also used 3 opencore designs [26]: Simple_spi which is a SPI core, Biquad which is a DSP core, and Ac-97, a controller core. All 3 are written in Verilog. These data sets were also used many times in articles on Design and Test.

Table 1 describes these benchmarks. The first line gives the name of the circuit (b09,..., b19) for ITC 99 and three OpenCore (single-spi, biquad, Ac-97). The following lines give the number of memory elements number of FF and the RTL description language (VHDL or Verilog) for each design.

Since our optimization criterion is wirelength, and since we consider congestion during routing an important issue, experimentations were conducted till the place and route step. Two flows were considered, according to Figure 1. In both cases, synthesis, placement and routing were done by Magma's Talus integrated flow. In the case of the gate-level scan insertion flow, the scan chain was also inserted by Talus. No effort options were set for gate-level scan insertion, since

Table 1: Designs description.

Design	No. of FF	Language
b09	28	vhdl
b10	17	vhdl
b11	31	vhdl
b12	121	vhdl
b13	53	vhdl
b14	245	vhdl
b15	449	vhdl
b17	1415	vhdl
b18	3320	vhdl
b19	6642	vhdl
Simple-Spi (SS)	132	verilog
Biquad	204	verilog
Ac-97	2289	verilog

Talus does not offer any. Besides those flows, the designs scanned at the RT level were also fed into Tetramax, in order to check fault coverage rates.

5.3. Numerical Results. Table 2 presents results of the first phase (extraction graphs $G_D = (V_D, E_D)$ and $G_p = (V_p, E_p)$). The columns 1 and 2 give the name of circuits tested (*Design*) and the size in number of memory elements (no. of FF). The following columns give the size of the design graph G_D respectively the number of vertices $|V_D|$ (column 3) and the number of edges $|E_D|$ (column 4). Columns 5 and 6 present the parameters of the proximity graph G_P with the number of vertices $|V_P|$ and the number of edges $|E_P|$. We observe that the number of vertices $|V_P|$ is equal to the number of memory elements of the design (column 2). The last column gives the CPU time for the extraction of the graph G_P from the graph G_D (Times).

The computation of the graph G_P from G_D (cf. Section 4.1.3) involves the threshold parameter t. The practical value of t has been determined empirically. It has to be big enough in order not to put arbitrary limitations on the choices of the algorithm; and it has to be small enough for the design-graph not to fill the whole memory of the computer. For all designs that we tested, the value $t = 4$ was a suitable choice according to those two criteria. This value was used in all our experiments.

The figures for both wirelength after place and route, and computation times, have been gathered in Table 2. The column number of SC gives the number of scan chains for each design. In the wirelength column, GLS denotes the flow with gate-level scan; RTL-scan gives the values for the RTL-scan flow. The slack column gives the ratio (in percent) by which RTL-scan wirelength exceeds that for gate-level scan.

Computation times are given only for the two steps of the stitching optimization method, discarding the time for parsing and elaboration. The column G_P gives the time for constructing the proximity-graph; TSP is the time for the 2-approximation algorithm for the TSP.

We base our analysis of the comparison of wirelength between gate-level and RTL scan on the slack column of Table 3, since the absolute values cannot be directly interpreted. The slack varies between −23% (meaning RTL-scan wirelength is shorter) and 6%. A closer analysis shows that gate-level scan is better mainly for small designs; our method is always better for the bigger designs, with only negative values of slack for all designs with more than 300 FFs.

Table 4 gives the fault coverage obtained by Tetramax for the designs scanned at the RT-level. Let us note that all values are above 98%, and only two lay below 99.8%. We conclude that RTL scan insertion cannot degrade fault coverage too severely as compared to traditional scan.

Computation times are mainly dominated by the setup of the graph G_P; at worst, the whole optimization process takes a little more than 5 minutes, in the case of the b19 design which has 6000 FFs.

6. Conclusion and Perspectives

After giving some motivations for inserting scan at the RT-level, we have exposed what we believe is an important challenge for RTL-scan, which is finding a good stitching in one single pass, working only at the RT-level. Therefore, the purpose of the present work is to solve the following problem: given an RT-level description of the design (Verilog or Vhdl) we have to find a stitching ordering of the memory elements in the scan chain, which minimizes the impact of test circuitry on chip performance (area, power) and testing time. Then, we have motivated our choice of selecting wire length as the prime parameter to optimize.

To solve this problem, we proposed a new scan stitching algorithm for optimal scan chain insertion at RTL. The techniques used are derived from the combinatorial optimization and operations research domains. Indeed, our algorithm is divided into two main steps. The first step proposes a mathematical model describing the electronic problem. The second one offers a resolution methodology.

The model we propose is based on graph theory. To build it, we extract from the RTL description information on the proximity of the memory elements (existing paths between flip-flops, clock domains, and various other relations extracted from hierarchical analysis) and translate them in two graphs G_D and G_P using the Limited-BFS algorithm. Then, we have shown that the scan stitching decision problem is equivalent to the Traveling Salesman Problem and therefore *NP-Complete*. However, the 2-approximation for the TSP can be used to find near optimal solutions in polynomial time. From this, we developed our stitching algorithm.

Finally, we integrated our tool in an industrial design flow and performed experiments over several academic and industrial design benchmarks. Numerical evidence showed that we are able to limit such a cost due to scan insertion in a reasonable computing time, without impacting DFT quality, especially fault coverage. The method seems better than the traditional one for middle-sized designs. The industrial interest for our algorithms and tools is confirmed by our industrial partners. Our RT-level scan optimization algorithm has been incorporated into the tool HiDFT-SIGNOFF by DeFacTo Technologies. In case new flip-flops

Table 2: Characteristics of the graphs G_D and G_P.

Design	No. of FF	$\lvert V_D\rvert$	$\lvert E_D\rvert$	$\lvert V_P\rvert$	$\lvert E_P\rvert$
b09	28	401	719	28	378
b10	17	525	947	17	136
b11	31	1094	1845	31	465
b12	121	3879	6993	121	7260
b13	53	633	1042	53	1378
b14	245	27269	46778	245	29890
b15	449	33179	56798	449	100576
b17	1415	100358	171759	1415	697990
b18	3320	271320	462326	3320	2330318
b19	6642	531161	906201	6642	7061713
Simple-Spi	132	1427	8646	132	8778
Biquad	204	357	20706	204	20910
Ac-97	2289	18727	2381495	2289	707911

Table 3: Wirelengths and insertion times.

Design	No. of SC	Wirelength		Slack (%)	Insertion time (ms)	
		GL-S	RTL-S		P-graph	TSP
ITC 99 Benchmarks (VHDL)						
b09	2	2.06	1.63	−20.59	<1	<1
b10	2	1.94	2.06	5.97	10	<1
b11	3	4.77	5.03	5.38	20	10
b12	10	12.6	12.92	2.59	200	20
b13	5	3.39	3.32	−2.25	20	<1
b14	24	63.64	64.2	0.96	3 s	110
b15	24	126.3	122	−3.39	6 s	320
b17	70	395.6	370.4	−6.37	30 s	3 s
b18	300	1187.2	922.6	−22.29	3 m	10 s
b19	600	2329.6	1858.2	−20.24	5 m	20 s
Opencore designs (Verilog)						
Simple-Spi	10	11.5	10.4	−8.95	100	20
Biquad	20	34.1	31.3	−8.14	350	80
Ac-97	200	247.7	238.5	−3.71	30 s	8 s

Table 4: Fault coverage for RTL scan.

Design name	Fault coverage
ITC 99 Benchmarks (VHDL)	
b09	99.86%
b10	99.85%
b11	99.93%
b12	99.97%
b13	99.92%
b14	99.99%
b15	99.97%
b17	99.47%
b18	99.81%
b19	99.81%
Opencore designs (Verilog)	
Simple Spi	98.35%
Biquad	99.96%
Ac-97	99.80%

are added or removed to existing scan chains, it is important to goldenize the RTL code and reflect such changes directly into the RTL code. The DeFacTo tool HiDFT-SIGNOFF allows that by introducing the new scan architecture and by including the flip-flops new ordering. In this way, our work represents a progress in the state of the art, as previous works are not automated and/or were evaluated only for small designs.

Last but not least, an important contribution of our work is to make a neat separation between the DFT problem and the mathematical model. This separation allows the same software to work for several successive technology nodes.

Our initial results give rise to a number of new directions for further research. These are summarized below. First, one could investigate whether cumulating local and global optimization in the manner of [8] would improve our results. Another aspect that could be added to our model is the question of power during testing: can we take it into account without degrading ou results? Finally, design analysis at a

higher-level gives an opportunity to reduce both area overhead and test time application by adopting partial scan instead of full scan [16–18]. Deciding whether the combination of design analysis with our method could reduce even more the cost of scan would be worthwhile.

References

[1] B. Korte, L. Lovasz, H. J. Promel, and A. Schrijver, *Paths, Flows, and VLSI-Layout*, Springer, New York, NY, USA, 1990.

[2] S. H. Gerez, *Algorithms for VLSI Design Automation*, John Wiley & Sons, Chichester, UK, 1998.

[3] L. T. Wang, Y. W. Chang, and K. T. Cheng, *Electronic Design Automation: Synthesis, Verification, and Test*, Morgan Kaufmann, Boston, Mass, USA, 2009.

[4] Semiconductors Industry Association, *Test and Test Equipment. Update*, International Technical Roadmap for Semiconductor, 2010.

[5] L. T. Wang, C. E. Stroud, and N. A. Touba, *System-on-Chip Test Architectures: Nanometer Design For Testability*, Morgan Kaufmann, Boston, Mass, USA, 2008.

[6] S. Roy, "RTL based scan BIST," in *Proceedings of the VHDL International Users' Forum (VIUF '97)*, pp. 117–121, October 1997.

[7] C. Aktouf, H. Fleury, and C. Robach, "Inserting scan at the behavioral level," *IEEE Design and Test of Computers*, vol. 17, no. 3, pp. 34–42, 2000.

[8] Y. Huang, C. C. Tsai, N. Mukherjee, O. Samman, W. T. Cheng, and S. M. Reddy, "Synthesis of scan chains for netlist descriptions at RT-Level," *Journal of Electronic Testing*, vol. 18, no. 2, pp. 189–201, 2002.

[9] M. Hirech, J. Beausang, and X. Gu, "New approach to scan chain reordering using physical design information," in *Proceedings of the IEEE International Test Conference (ITC '98)*, pp. 348–355, October 1998.

[10] D. Berthelot, S. Chaudhuri, and H. Savoj, "An efficient linear time algorithm for scan chain optimization and repartitioning," in *Proceedings of the International Test Conference (ITC '02)*, pp. 781–787, Washington, DC, USA, October 2002.

[11] P. Gupta, A. B. Kahng, and S. Mantik, "Routing-aware scan chain ordering," *ACM Transactions on Design Automation of Electronic Systems*, vol. 10, no. 3, pp. 546–560, 2005.

[12] P. Gupta, A. B. Kahng, I. I. Măndoiu, and P. Sharma, "Layout-aware scan chain synthesis for improved path delay fault coverage," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 7, pp. 1104–1114, 2005.

[13] B. B. Paul, R. Mukhopadhyay, and I. S. Gupta, "Genetic algorithm based scan chain optimization and test power reduction using physical information," in *Proceedings of the IEEE Region 10 Conference (TENCON '06)*, Hong Kong, November 2006.

[14] K. D. Boese, A. B. Kahng, and R. S. Tsay, "Scan chain optimization: heuristic and optimal solutions," Internal Report CS Department, University of California at Los Angeles, Los Angeles, Calif , USA, 1994.

[15] S. Makar, "Layout-based approach for ordering scan chain flip-flops," in *Proceedings of the IEEE International Test Conference (ITC '98)*, pp. 341–347, October 1998.

[16] S. Bhattacharya and S. Dey, "H-SCAN: a high level alternative to full-scan testing with reduced area and test application overheads," in *Proceedings of the 14th IEEE VLSI Test Symposium*, pp. 74–80, Princeton, NJ, May 1996.

[17] T. Asaka, S. Bhattacharya, S. Dey, and M. Yoshida, "H-SCAN+: a practical low-overhead RTL design-for-testability technique for industrial designs," in *Proceedings of the IEEE International Test Conference (ITC '97)*, pp. 265–274, Washington, DC, USA, November 1997.

[18] R. B. Norwood and E. J. McCluskey, "High-level synthesis for orthogonal scan," in *Proceedings of the 15th VLSI Test Symposium*, pp. 370–375, May 1997.

[19] Private communication.

[20] L. Zaourar and Y. Kieffer, "Scan chain optimization for integrated circuits testing : an operations research perspective," in *Proceedings of the 23rd European Conference on Operational Research (EURO '09)*, p. 289, Bonn, Germany, July 2009.

[21] L. Zaourar, Y. Kieffer, C. Aktouf, and V. Julliard, "global optimization for scan chain insertion at the RT-level," in *Proceedings of the IEEE Computer Society Annual Symposium VLSI (ISVLSI '11)*, pp. 321–322, University Pierre and Marie Curie, Paris, France, July 2011.

[22] L. Zaourar, Y. Kieffer, and C. Aktouf, "An innovative methodology for scan chain insertion and analysis at RTL," in *Proceedings of the Asian Test Symposium (ATS '11)*, pp. 66–71, IEEE Computer Society, Washington, DC, USA, November 2011.

[23] W. J. Cook, W. H. Cunningham, and W. R. Pulleyblank, *Combinatorial Optimization*, Wiley-Interscience, New York, NY, USA, 1997.

[24] http://glaros.dtc.umn.edu/gkhome/views/metis/.

[25] http://www.cerc.utexas.edu/itc99-benchmarks/bench.html.

[26] http://www.opencores.org/.