Routing-Aware Scan Chain Ordering

PUNEET GUPTA and ANDREW B. KAHNG
University of California at San Diego
and
STEFANUS MANTIK
Cadence Design Systems, Inc.

Scan chain insertion can have a large impact on routability, wirelength, and timing of the design. We present a *routing-driven* methodology for scan chain ordering with minimum wirelength objective. A routing-based approach to scan chain ordering, while potentially more accurate, can result in TSP (Traveling Salesman Problem) instances which are asymmetric and highly nonmetric; this may require a careful choice of solvers. We evaluate our new methodology on recent industry place-and-route blocks with 1200 to 5000 scan cells. We show substantial wirelength reductions for the routing-based flow versus the traditional placement-based flow. In a number of our test cases, over 86% of scan routing overhead is saved. Even though our experiments are, so far, timing oblivious, the routing-based flow also improves evaluated timing, and practical timing-driven extensions appear feasible.

Categories and Subject Descriptors: B.7.2 [**Integrated Circuits**]: Design Aids—*Layout*

General Terms: Algorithms, Design, Performance, Verification

Additional Key Words and Phrases: Layout, scan chain, testing

1. INTRODUCTION AND MOTIVATION

In VLSI design for testability, a *scan chain* is commonly used to connect the shift registers that store the input and output vectors during the testing phase of manufacturing. Registers in the scan chain are connected as a single path with ends of the path connected to a *primary input* (PI) pad and a *primary output* (PO) pad. Test input values are shifted into the registers through the PI pad; then, a test is performed and the test output values are shifted out through the PO pad. Figure 1 depicts a simple example of a scan chain.

Research at the University of California San Diego was supported by a grant from the MARCO Gigascale Silicon Research Center.
Authors' addresses: P. Gupta, ECE Department, University of California San Diego; email: puneet@ucsd.edu; A. B. Kahng, ECE and CSE Departments, University of California San Diego; email: abk@ucsd.edu; S. Mantik, Cadence Design Systems, Inc., San Jose, CA; email: smantik@cadence.com.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 1515 Broadway, New York, NY 10036 USA, fax: +1 (212) 869-0481, or permissions@acm.org.
© 2005 ACM 1084-4309/05/0700-0546 $5.00

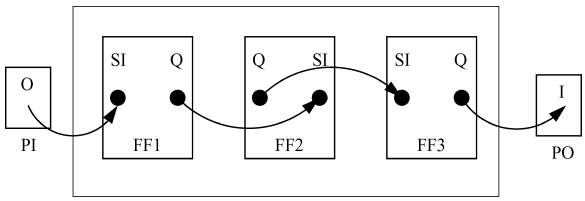

Fig. 1. Example of a scan chain with three scan registers *FF*1, *FF*2, and *FF*3. In each sequential cell, *SI* and *Q* denote the scan-in pin and scan-out pin. *PI* is the primary input pad, and *PO* is the primary output pad.

One of the primary objectives in design-for-testability is to minimize the impact of test circuitry on chip performance and cost. Thus, it is essential to minimize the wirelength of a scan chain: this decreases wiring congestion and/or reduces the chip area while, at the same time, increasing signal speed by reducing capacitive loading effects on nets that share register pins with the scan chain.

Previous placement-based scan chain ordering approaches compute the cost of stitching one flip-flop to another as either cell-to-cell Manhattan distance [Hirech et al. 1998; Makar 1998; Barbagello et al. 1996] or pin-to-pin Manhattan distance [Boese et al. 1994; Kobayashi et al. 1999]. The former metric gives a symmetric TSP, while the latter gives rise to an almost symmetric TSP [Boese et al. 1994]. The fundamental assumption in all current work on layout-driven scan chain ordering is that the wirelength overhead due to scan insertion is equal to the Manhattan distance between the scan-in and scan-out pins of the flip-flops. However, this assumption is incorrect: the scan connection need only reach the output *net*, not the output *pin*.

In this work, we propose a (trial) routing-based flow for scan chain ordering that uses the *incremental* routing cost (connecting to existing or anticipated routing, rather than to the output pin) as the cost measure for a scan connection. This is in contrast to existing placement-based methods which use simply the Manhattan distance from the flip-flop output pin to the scan-in pin of the other flip-flop as the cost measure. Under our formulation, the resulting Asymmetric Traveling Salesman Problem (ATSP) may be highly nonmetric. We give an efficient method to calculate the costs of the ATSP instance based on a trial routing of nonscan nets. Our work considers the possibility of using both Q and $\bar{Q}$ pins of the flip-flop to make any given scan connection, and it also extends to timing- and noise-driven scan chain ordering (in a more detailed routing-driven context).

In Figure 2, the existing routes are shown by solid lines, while potential scan connection routes are shown by dotted lines. We label the possible scan connection routes by their respective lengths, w, x, y, and z. Note that

—the cost of connecting the Q output of FF A to the *SI* pin of FF B (denote this by *AB*) is given by the length of the routing segment w which is much less than the total Manhattan distance between the corresponding pins. Thus, a placement-based cost metric will inaccurately estimate the cost of making this scan connection.

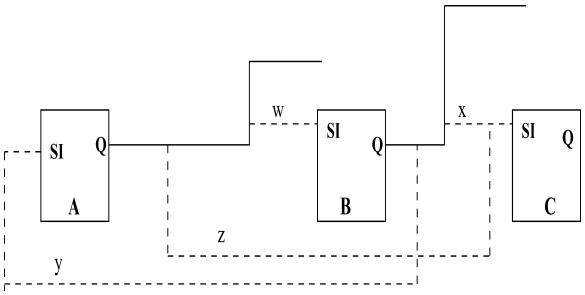

Fig. 2. An example showing the highly asymmetric and nonmetric nature of the ATSP when doing incremental scaninsertion.

—this formulation of the TSP can be highly asymmetric. For instance, in Figure 2, $BA(=y) \gg AB(=w)$.

—we can also have nonmetric TSP instances (i.e., the triangle inequality may not hold). In Figure 2, $AB(=w) + BC(=x) < AC(=z)$.

The scan chain order generated using routing information should always be at least as good as the placement-based order as it can more accurately reflect the wirelength costs of scan stitching.

In the remainder of this article, we discuss our scan chain ordering approach. Section 2 discusses related previous work. Section 3 describes our routing-based scan ordering approach. Experiments are described in Section 4, and extensions and conclusions are given in Section 5.

2. PREVIOUS WORK

In this section, we survey previous work in two relevant areas: layout-based scan chain ordering approaches and ATSP heuristics and solvers.

2.1 Layout-Based Scan Ordering

Several previous works have dealt with the problem of scan chain ordering based on layout information. These works achieve scan orderings that are essentially placement-based, typically by transforming the scan chain ordering problem to a symmetric or asymmetric TSP. Feuer and Koo [1983] wrote perhaps the first published work showing how TSP heuristics can be applied to scan chain optimization. They translate a given scan chain instance into a symmetric TSP instance, allowing symmetric TSP heuristics to be used without modification. However, the translation weakens the effectiveness of TSP heuristics because it creates highly irregular vertex distances. In essence, the instance loses its underlying geometry and becomes harder to optimize. Hirech et al. [1998], Makar [1998], and Barbagello et al. [1996] treat the problem as a symmetric TSP and use a simple nearest-neighbor heuristic which is based on the assumption that the triangle inequality holds. However, scan chain instances can be highly asymmetric and may not even remain metric. Boese et al. [1994] and Kobayashi et al. [1999] modify 2-opt and 3-opt TSP heuristics to

take into account the asymmetric nature of the scan chain problem. Lin et al. [1996] orders the scan chain after global routing but uses channel congestion as the edge cost measure which makes it applicable only in the channel routing context.

2.2 TSP Solvers

An excellent although slightly dated overview of the traveling salesman literature is contained in the 1985 book of Lawler et al. [1985]. More recent studies include a very comprehensive empirical comparison of symmetric TSP heuristics by Johnson [1990]. Since Johnson's study, works by Bentley [1992] and Reinelt [1992] have suggested ways to reduce the running times of the more successful heuristics, while Martin et al. [1991] have given an extension to earlier heuristics that improves the quality of the returned tour while increasing running times. The literature for asymmetric TSP is less extensive than for symmetric TSP and includes a heuristic by Kanellakis and Papadimitriou [1992] which modifies the Lin-Kernighan (LK) symmetric TSP heuristic. More recent studies by Miller and Pekny [1991] and Zhang [1992] have applied branch-and-bound to obtain optimal solutions to asymmetric TSP instances. Although branch-and-bound has exponential time complexity in the worst case, these two studies have shown that it can be efficient for a number of large ATSP instances.

Hong et al. [1997] reviews the use of large-step Markov chain (LSMC) methods which alternately apply (i) a (greedy) local optimization procedure *Descent*, followed by (ii) a "kick move" which perturbs the current local minimum solution in order to obtain a starting solution for the next *Descent* application. Local search (i.e., *Descent*) procedures used in implementations include LK as well as k-opt methods [Martin et al. 1991] used in both LK and a fast implementation of 3-opt; Johnson [1990] used LK only. Kick a move perturbation of the current local minimum tour is typically achieved using a k'-change with k' not necessarily equal to k. Both Martin et al. [1991] and Johnson [1990] use random double-bridge 4-change kick moves. According to Martin et al. [1991], the double-bridge kick move is chosen for its ability to produce large-scale changes in the current tour without destroying the solution quality via too large a random perturbation. The large-step Markov chain (LSMC) heuristic of Martin et al. [1991] and the iterated LK heuristic of Johnson [1990] are believed to be the best performing iterated descent variants (and, indeed, the best performing of all heuristics for obtaining near-optimal solutions Johnson and McGeoch [1997]). Boese et al. [1994] introduced restricted 2-opt and 3-opt moves to preserve directionality for solving an ATSP; their *ScanOpt* code, an implementation of the LSMC heuristic with restricted moves for ATSP, is available on the web at http://www.gigascale.org/bookshelf/Slots/ScanOpt/. Finally, results of the 8th DIMACS Implementation Challenge http://www.research.att.com/dsj/chtsp/. for TSP are available in Johnson et al. [2002]. The method of Helsgaun [2000] is shown to give the best tours if running time is not a constraint.

As suggested in Figure 2, asymmetry of the TSP will increase with use of any routing-based distance measure, while metricity decreases. To assess the impact of the routing-based TSP distance metric on the nature of the resultant

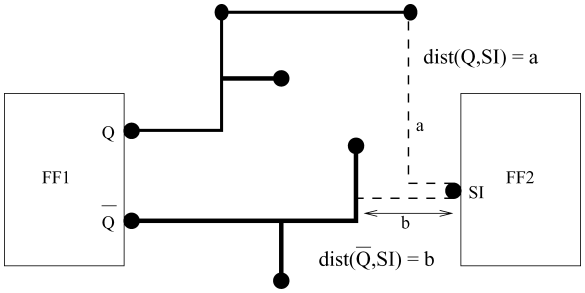

Fig. 3. Pin-to-Net distance measurement for routing-driven scan ordering.

TSP instances, we may adopt the asymmetry and metricity measures presented in Johnson et al. [2002]. The asymmetry measure is used to measure the extent to which the distance matrix d departs from symmetry, while the metricity measure is used to measure the extent to which the distances violate the triangle inequality.[1]

3. ROUTING-AWARE SCAN CHAIN ORDERING

In this section, we describe our routing-aware approach to scan chain ordering with minimum wirelength. We consider a purely wirelength-driven approach. A timing-aware extension which takes into account timing slacks at all relevant sinks as well as available buffer locations will be sketched as a possible future extension in the last section of this article.

The cost of stitching one flip-flop after another is determined by the estimate of minimum wirelength required to make the connection. This is obtained by finding the minimum Manhattan distance from the routed net driven by the first flip-flop's scan-out pin (Q or $\bar{Q}$) to the second flip-flop's scan-in pin (SI). Let $dist(Q, SI)$ ($resp.$, $dist(\bar{Q}, SI)$) denote the length of the best possible Manhattan route for adding the SI pin of flip-flop ff_{in} (e.g., $FF2$) to the fanout tree of the $Q(\bar{Q})$ output of flip-flop ff_{out} (e.g., $FF1$). Then, we compute the cost of placing scan flip-flop ff_{out} immediately before ff_{in} as $min(dist(Q, SI), dist(\bar{Q}, SI))$ (see Figure 3). The value of $dist(Q, SI)$ (similarly for $dist(\bar{Q}, SI)$) is calculated as follows: $dist(Q, SI) = min_k(Manhattan\ \ distance(segment_k, ff_{in} \rightarrow SI))$ where $segment_k, k = 1, 2, \ldots,$ are all routed segments on the fanout tree of the Q output of ff_{out}. For each pair of ff_{out} and ff_{in} we calculate the scan cost, yielding a cost matrix that will be used by the ATSP solver. Once we obtain the tour from the ATSP solver, we update the netlist and reroute the design from scratch, that is, the routing starts from an unrouted instance. Note that our flow is not

[1]For the asymmetry measure, we use the ratio of average value of $|d(v_i, v_j) - d(v_j, v_i)|$ to average value of $|d(v_i, v_j) + d(v_j, v_i)|$ where $d(v_i, v_j)$ denotes the distance of vertex v_j from vertex v_i. This quantity is 0 for symmetric matrices and has a maximum value of 1. For the metricity measure, we first compute for each pair of distinct vertices v_i, v_j, $d'(v_i, v_j) = min(d(v_i, v_j), min(d(v_i, v_k) + d(v_k, v_j))) : 1 \leq k \leq N$ where N is the total number of vertices. The metric is the average over all pairs v_i, v_j of $(d(v_i, v_j) - d'(v_i, v_j))/d(v_i, v_j)$. The instance obeys the triangle inequality if and only if we get a value of 0. For nonnegative distances, the above quantity has a maximum value of 1. We call this measure *nonmetricity* since it increases with decreasing metricity. These measures are given in Johnson et al. [2002].

an ECO flow.[2] Kahng and Mantik [2000] demonstrate that incremental routing can achieve lower solution quality than from-scratch routing and that the magnitude of change in this instance can strongly affect the resulting quality of the result. Thus, we use the from-scratch routing for a better solution despite what is essentially a doubling of CPU cost. Our flow is basically the same as the traditional scan-reordering flow but is driven by global routing or even trial-detailed routing; our contribution lies in showing its practicality as well as its surprisingly large reductions in scan overhead.

4. EXPERIMENTS AND RESULTS

In this section, we describe our simulation setup and the experimental test cases used. We are interested in understanding the implications of routing-based scan ordering on the choice of TSP solver and on the potential layout quality improvement. The layout quality measures are

—total wirelength,
—number of routing violations,
—total router runtime,
—total wirelength due to scan routing,
—number of timing violations (paths with negative slack), and
—worst slack.

We use Cadence *Silicon Ensemble v*5.3.125 (SE) and Cadence *QPlace v*5.1.68 as the physical design tool to perform the industry placement-based scan chain ordering. In addition, we use Cadence *WRoute v*2.2.31 as the routing tool. We have developed basic utilities for extracting the industry tool's scan ordering from a routed DEF (the order is not otherwise available in the output DEF), for generating pin-to-pin distances from the placed DEF, for generating minimum pin-to-net distances from the routed DEF, and for plugging a solver-generated scan order into DEF for routing.

4.1 Flows

The basic elements of the flows are given as follows.

(1) *Initial QPlace.* The design is placed with QPlace to generate a placed DEF netlist. There are two placement variations that we can generate:
   (a) *NonTiming Driven (NTD).* When the timing library is not supplied, QPlace will optimize the placement based on wirelength (and possibly congestion).
   (b) *Timing Driven (TD).* When given a timing library and timing constraints, QPlace will try to optimize the timing (i.e., minimize the incidence of negative slacks).

[2]ECO (Engineering Change Order) is an incremental type of flow that uses a prerouted design as the input. In such a flow, the router will update the routing while trying to preserve the previous solution as much as possible.

(2) *Placement-Based Scan Order*
   (a) *SE.* SE is used to attach scan to the placed netlist. The scan order is implicitly generated by SE using, as we understand, a greedy heuristic followed by iterative (hill-climbing) improvement.
   (b) *QP-Scan.* QPlace is used to attach scan to the placed netlist. As we understand, the QP scan ordering is more recent (dating from approximately early 2000) and gives superior results to the SE scan ordering; it uses a k-opt type of iterative improvement.
   (c) *Our (ScanOpt, LKH).* We extract scan flip-flop locations from the placed DEF. We compute pairwise pin-to-pin distances to construct the TSP cost matrix. The ATSP solver (ScanOpt, LKH) is then used to obtain a scan chain order. This order is incorporated into the placed netlist by adding the scan-in pin to (i.e., as an additional sink of) the existing net of the scan-out pin and by adding a new net for any scan-out pin that does not have any previous connection (the scan-out scan-in pair follows the order specified by the ATSP solver).

(3) *WRoute.* The placed netlist is routed using WRoute. Similar to the variations in placement, there are also two variations in the routing.
   (a) *NonTiming Driven Routing*
   (b) *Timing Driven Routing*

(4) *Routing-Based Scan Order*
   (a) *SE.* SE is used to attach scan to the routed netlist.
   (b) *QP-Scan.* QPlace is used to attach scan to the routed netlist.
   (c) *Our (ScanOpt).* We extract fanout routing trees of all the scan flip-flops from the routed netlist. The ATSP cost matrix is computed from the minimum pin-to-tree distances. The ATSP solver then computes the routing-based scan order.

The previously mentioned steps are used to construct the following scan chain insertion flows.

—*Flow Base-NTD: 1A, 3A.* Baseline NTD total wirelength with no scan nets.
—*Flow Base-TD: 1B, 3B.* Baseline TD total wirelength and timing with no scan nets.
—*Flow PBSE-NTD: 1A, 2A, 3A.* NTD placement-based SE flow.
—*Flow PBSE-TD: 1B, 2A, 3B.* TD placement-based SE flow.
—*Flow PBQP-NTD: 1A, 2B, 3A.* NTD placement-based QP-Scan flow.
—*Flow PBQP-TD: 1B, 2B, 3B.* TD placement-based QP-Scan flow.
—*Flow PBOur-NTD: 1A, 2C, 3A.* Our NTD placement-based flow. This directly compares with Flow PBSE-NTD and Flow PBQP-NTD, that is, industry vs. our placement-based scan ordering solvers. The comparison is clouded by the possibility that the industry tool's edge costs and objective function may be different from ours.
—*Flow PBOur-TD: 1B, 2C, 3B.* Our TD placement-based flow. Similarly, this directly compares with Flow PBSE-TD and Flow PBQP-TD.

Table I. A Comparison of ATSP Solvers

	Tour Cost (μm)		Running Time (sec.)	
Test Case	ScanOpt	LKH	ScanOpt	LKH
A (pin-to-pin)	21609	20632	1441	5670
A (pin-to-net)	9297	7511	2149	2717

—*Flow RBSE-NTD: 1A, 3A, 4A, 3A.* NTD routing-based SE flow.

—*Flow RBSE-TD: 1B, 3B, 4A, 3B.* TD routing-based SE flow.

—*Flow RBQP-NTD: 1A, 3A, 4B, 3A.* NTD routing-based QP-Scan flow.

—*Flow RBQP-TD: 1B, 3B, 4B, 3B.* TD routing-based QP-Scan flow.

—*Flow RBOur-NTD: 1A, 3A, 4C, 3A.* Our NTD routing-based flow. This directly compares with Flow RBSE-NTD and Flow RBQP-NTD, i.e., industry vs. our routing-based scan ordering solvers. Again, the comparison is clouded by possible differences in costing and objective function. Flow RBOur-NTD also compares to Flow PBOur-NTD to assess the impact of routing-based ordering vis-a-vis placement based ordering.

—*Flow RBOur-TD: 1B, 3B, 4C, 3B.* Our TD routing-based flow. Similarly, this directly compares with Flow RBSE-TD and Flow RBQP-TD.

We extracted scan chain orders from the industry tool's flow. Scan chain orders generated by SE in Flow PBSE-NTD(-TD) and Flow RBSE-NTD(-TD) are identical. Similarly, scan chain orders generated by QPlace in Flow PBQP-NTD(-TD) and Flow RBQP-NTD(-TD) are identical. We interpret this as the absence of any routing-based ordering in both SE and QPlace. Therefore, Flows RBSE-NTD, RBSE-TD, RBQP-NTD, and RBQP-TD become redundant and we do not report results for them. In total, there are ten distinct results for each test case, corresponding to Flows Base-NTD, Base-TD, PBSE-NTD, PBSE-TD, PBQP-NTD, PBQP-TD, PBOur-NTD, PBOur-TD, RBOur-NTD, and RBOur-TD.

4.2 ATSP Solver Comparison

As our results heavily depend on the quality of the TSP tour, we compare two ATSP solvers *ScanOpt* and *LKH-1.2* [Helsgaun 2000]. *LKH-1.2* is reported to be one of the best ATSP solvers currently available [Johnson et al. 2002]. *LKH* converts an ATSP instance to a symmetric TSP instance by doubling the number of cities. This can cause huge runtimes for large ATSP instances. We use *ScanOpt* as our TSP solver as its running time is reasonable even for very large TSP instances (more than 10000 cities). A comparison of *ScanOpt* and *LKH* for the smallest of our test cases (i.e., *A* with 1226 cities) is given in Table I. With two different measures (tour cost and CPU time) in test case A, *ScanOpt* is more favorable with respect to running time, while *LKH* may give a better tour with some extra runtime.[3]

4.3 Test Cases

We consider three test cases obtained from industry sources. Each of the test cases was obtained in LEF/DEF format (http://www.openeda.org/) and then

[3]For practical reasons, we use *ScanOpt* for our experiments.

Table II. Characteristics of the Test Cases

Test Case	# Cells	# Scan FFs	# Scan Chains	Die Area mm^2	# Metal Layers
A/A_{swap}	6390	1226	2	0.526	4
A_{expand}	6390	1226	2	0.632	4
B/B_{swap}	40350	1975	1	6.875	4
B_{expand}	40350	1975	1	8.373	4
C/C_{swap}	34235	4550	10	3.846	4
C_{expand}	34235	4550	10	5.611	4

Table III. Asymmetry and (Non)Metricity Measures for the Test Cases

Non-TD Instance	Asymmetry/NonMetricity		
	Cell-to-Cell	Pin-to-Pin	Pin-to-Net
A	0/0	0.0122/0.0975	0.1199/0.6356
A_{swap}	0/0	0.0122/0.0975	0.1308/0.6959
A_{expand}	0/0	0.0105/0.1079	0.1287/0.6447
B	0/0	0.0122/0.2044	0.0254/0.3628
B_{swap}	0/0	0.0122/0.2044	0.0424/0.6322
B_{expand}	0/0	0.0108/0.1898	0.0218/0.3787
C	0/0	0.0039/0.1695	0.0484/0.5978
C_{swap}	0/0	0.0039/0.1695	0.0555/0.7140
C_{expand}	0/0	0.0037/0.2228	0.0568/0.5932
TD	Cell-to-Cell	Pin-to-Pin	Pin-to-Net
A	0/0	0.0121/0.0904	0.1187/0.6549
A_{swap}	0/0	0.0121/0.0904	0.1295/0.7134
A_{expand}	0/0	0.0111/0.1025	0.1316/0.6733
B	0/0	0.0121/0.2537	0.0276/0.4297
B_{swap}	0/0	0.0121/0.2537	0.0491/0.6426
B_{expand}	0/0	0.0111/0.1523	0.0284/0.3440
C	0/0	0.0039/0.2185	0.0598/0.6098
C_{swap}	0/0	0.0039/0.2185	0.0683/0.7238
C_{expand}	0/0	0.0037/0.2228	0.0568/0.5932

modified to merge its multiple scan chains into one scan chain. We then generated alternate placements for each test case by either (1) relaxing the site map by 20% or (2) randomly swapping some scan flip-flop locations.[4] The basic parameters of the test cases are given in Table II. A_{swap} denotes the test case with placement of A altered by randomly swapping the placements of scan flip-flops, while A_{expand} denotes the test case obtained by increasing the site map of A. The asymmetry and nonmetricity values for the corresponding TSP instances are shown in Table III. Note that nonmetricity values increase markedly (i.e., metricity decreases) with the new pin-to-net distance measure. This supports the need to use an ATSP solver in our scan ordering approach.

4.4 Placement vs. Routing-Driven Ordering

Table IV shows the total wirelength after detailed routing for the baseline Flow Base-NTD(-TD) and scan overhead for each of the remaining flows described

[4]The number of swaps differed for each design to ensure routability. For test cases A, B, and C, 30 swaps, 50 swaps and 50 swaps respectively were made. From this, it can be seen that the initial placed instances are already fairly tight in terms of routability.

Table IV. Post-Detailed Routing Wirelength and Scan Overhead for Various Scan Chain Insertion Flows

Test Case	Total WL (μm) Base-NTD	Scan Overhead(μm) PBSE-NTD	PBQP-NTD	PBOur-NTD	RBOur-NTD
A	901377	38464	20550	16595	6230
A_{swap}	947229	31727*	20590	14383	9128*
A_{expand}	925623	49390	30596	23520	14229
B	4145339	146316	73356	63483	54527
B_{swap}	4554848	144622	75967	65385	49690
B_{expand}	4687923	155366	76370	72976	60731
C	8467723	334200	164384	131491	85724
C_{swap}	9078337	327720	166846	148106	98552
C_{expand}	8957890	416030	205973	180373	127850
	Base-TD	PBSE-TD	PBQP-TD	PBOur-TD	RBOur-TD
A	864765	42280*	21502*	21939	6540*
A_{swap}	925899*	39202*	17241*	14457*	6798*
A_{expand}	950629	51421	34091	28433	14138
B	4004035	148707	72811	67369	55353
B_{swap}	4510690	147672	73268	65375	49268
B_{expand}	4296941	157460	80282	76911	63329
C	8250811	338108	152479	135496	79540
C_{swap}	8987769	352032	177312	136687	83495
C_{expand}	8957890	293708	116033	86851	41974

*Indicates that the routing completed with violations.

in Section 4.1. Scan overhead is computed as the difference between baseline wirelength and post-scan-insertion wirelength. The routing in all the cases is wirelength-driven rather than timing-driven. We also retain the same placement for all the flows (including the no-scan case) to have a better comparison. CPU time of all the routing runs are shown in Table V (CPU times are normalized to a 143Mhz SUN Ultra-1). CPU time for Flow RBOur-NTD and RBOur-TD are the sum of the initial routing time (before scan-insertion) and the final routing time (after the scan-insertion). Table VI shows the effect on timing violations and slacks. We use setup violations for the timing measurements.

Table IV clearly shows that both Flow RBOur-NTD and RBOur-TD produce better wirelength than Flow PBOur-NTD and Flow PBOur-TD, respectively. This gives an unbiased comparison of the placement-based ordering and the routing-based ordering. Flows RBOur-NTD and RBOur-TD also have better wirelength than the industry flows, Flow PBSE-NTD, PBSE-TD, PBQP-NTD, and PBQP-TD. The reduction in wirelength ranges from 20.5% to 85.7%. This shows that, by exploiting extra (trial) routing information, we can indeed achieve better solutions.

Furthermore, we may observe that QP-Scan (Flow PBQP-NTD(-TD)) clearly has a better TSP solver than SE (Flow PBSE-NTD(-TD)), while our ATSP solver (Flow PBOur-NTD(-TD)) is as good as QP-Scan.[5] There are three cases where Flow RBOur-NTD(-TD) completes the routing with violations; however, with these cases, the industry flow has violations as well.

[5]Here it is important to note that Flow PBOur-NTD(-TD) and Flow RBOur-NTD(-TD) have the additional freedom of choosing between Q and $\bar{Q}$ to connect the SI pin.

Table V. Total Router CPU Times (in Seconds) for Various Scan Chain Insertion Flows Normalized to SUN Ultra-1 at 143MHz (CPU times for Flow RBOur-NTD and RBOur-TD are the sum of the initial routing time (before scan-insertion) and the final routing time.)

Test Case	Base-NTD	PBSE-NTD	PBQP-NTD	PBOur-NTD	RBOur-NTD
A	881	718	1271	915	1455
A_{swap}	733	3562	2975	2555	3265
A_{expand}	326	1317	366	343	666
B	1035	1058	1050	1058	2091
B_{swap}	1076	1087	1082	1086	2155
B_{expand}	1137	1153	1141	1145	2281
C	4028	4513	4189	4157	8156
C_{swap}	8549	6922	6271	6180	14539
C_{expand}	2570	2904	2731	2680	5232
Test Case	Base-TD	PBSE-TD	PBQP-TD	PBOur-TD	RBOur-TD
A	1927	4681	2821	2708	4713
A_{swap}	3641	9315	6440	5653	8461
A_{expand}	701	684	651	645	1334
B	2582	2403	2396	2388	4976
B_{swap}	2449	2501	2493	2512	4918
B_{expand}	2432	2488	2468	2480	4905
C	3639	5907	5586	5514	9132
C_{swap}	6977	9456	7939	7813	14448
C_{expand}	2557	6106	5930	5930	8387

Table VI. Timing Results (Min Slack and Number of Setup Time Violations) for the TD Scan-Insertion Flows

	Min Slack (ns)/# Setup Time Violations			
Test Case	PBSE-TD	PBQP-TD	PBOur-TD	RBOur-TD
A	5.31/624	5.24/602	5.34/586	5.50/578
A_{swap}	7.62/764	7.57/722	7.17/719	7.50/744
A_{expand}	5.78/817	6.02/817	5.92/777	6.26/781
B	5.26/4	5.25/4	5.26/4	5.26/4
B_{swap}	5.27/5	5.27/5	5.27/5	5.27/5
B_{expand}	5.23/4	5.23/4	5.23/4	5.23/4
C	15.26/6121	15.63/5954	15.17/5816	14.99/5725
C_{swap}	24.58/6521	24.38/6418	23.10/6170	24.32/5973
C_{expand}	25.69/7328	26.38/7398	26.46/7458	24.60/7320

In the timing domain (Table VI), although our current flow does not consider slacks in the cost matrix calculation, we see a reduction in the number of timing violations. In addition, the magnitude of the timing violations (i.e., negative slacks) are not worse than the industry flows in some cases.

4.5 Impact of Freedom in Output Pin Choice

Using the unused output pin of the scan flip-flop to make the scan connection is one of the ways in which industrial flows try to keep the timing impact of scan insertion under control. We have noticed that up to 60% of the scan nets fall into this category in some industrial benchmark designs. This puts unnecessary constraints on design and synthesis besides increasing the total wirelength overhead of the scan insertion. In this section of the article, we assess the

Table VII. Impact of Constraining the Output Pin for Scan Connections

Test Case	WL Overhead			
	RBOur-NTD-(i)	RBOur-NTD-(ii)	RBOur-TD-(i)	RBOur-TD-(ii)
A	6230	32348	6540*	21623*
A_{swap}	9128*	34794	6798*	18441*
A_{expand}	14229	31129	14138	29820
B	54527	73678	55353	68461
B_{swap}	49690	84391	49268	65631
B_{expand}	60731	76712	63329	69658
C	85724	184661	79540	138308
C_{swap}	98552	217655	83495	157653
C_{expand}	127850	171448	41974	81311

wirelength impact of constraining the output pin for scan connection. We run the routing driven Flow RBOur-NTD(-TD), first with unconstrained output pin (Flow RBOur-NTD/TD-(i)), and then with a preconstrained output pin (Flow RBOur-NTD/TD-(ii)).

Table VII shows that constraining the output pin results in a 10%–420% increase in the wirelength overhead of scan insertion. Routing-driven scan chain ordering achieves good timing (as shown in Table VI) and preconstraining the scan-out pin is unnecessary and an overconstraint.

4.6 Impact on Hold Time Violations

Typically, adding scan connections reduces the number of hold time violations on the design pins if the connections are pin-to-net due to the added load and the consequential increased delay on the sink pins. However, in a minimum wirelength scan chain ordering context, some scan connections can be very short and hence cause hold time violations on the scan-in pin itself.

Placement-based scan chain ordering seeks minimum wirelength pin-to-pin connections. Using the unused output pin to make the scan connection is important for placement-based ordering since the connection is chosen based on pin-to-pin distances and may be very poor in terms of actual routing distance. In the case of routing-based ordering, fixing the output pin for scan connection is an overconstraint. Since the connection choice is based on pin-to-net distances, the unconstrained routing-based ordering achieves almost the same timing quality (as is obvious from results in Table VI) as constrained placement-based ordering. Placement-based ordering can, therefore, result in very short pin-to-pin scan nets. Pin-to-net connections, though short, have larger delays due to loading of the net by sinks other than the scan-in pin. Thus, placement-based ordering is more prone to hold time violations on the scan-in pin than the routing-based approach.

To establish variant test cases that can confirm the above hypothesis, we modify the TLF timing libraries in our designs to increase the hold time of all flip-flops. The results of the hold time check using *Cadence Pearl* are given in Table VIII.

From the results in Table VIII, it is clear that the routing-driven flow is not worse than the other flows in terms of number of hold time violations or hold

Table VIII. Hold Time Results (Min Slack and Number of Hold Time Violations) for Scan-Insertion Flows

	Min Slack (ns)/# Hold Time Violations			
Test Case	PBSE-NTD	PBQP-NTD	PBOur-NTD	RBOur-NTD
A	2.5/159	2.5/210	2.5/122	2.5/72
A_{swap}	2.5/156	2.5/213	2.5/119	2.5/69
A_{expand}	2.5/149	2.5/206	2.5/100	2.5/57
B	3.4/9962	3.2/9973	3.2/10024	3.2/10037
B_{swap}	3.3/9979	3.2/9948	3.2/10001	3.2/10002
B_{expand}	3.3/10098	3.3/10088	3.3/10049	3.3/10063
C	3.6/20	3.6/22	3.6/21	3.6/20
C_{swap}	3.6/21	3.6/21	3.6/21	3.6/20
C_{expand}	3.6/22	3.6/22	3.6/20	3.6/20
Test Case	PBSE-TD	PBQP-TD	PBOur-TD	RBOur-TD
A	2.5/158	2.5/207	2.5/108	2.5/43
A_{swap}	2.5/159	2.5/212	2.5/112	2.5/39
A_{expand}	2.5/151	2.5/201	2.5/130	2.5/64
B	3.2/9979	3.4/10056	3.4/10038	3.4/10080
B_{swap}	3.3/9955	3.4/9989	3.4/9997	3.4/9982
B_{expand}	3.4/10128	3.4/10112	3.4/10110	3.4/10102
C	3.6/22	3.6/25	3.6/20	3.6/20
C_{swap}	3.6/21	3.6/22	3.6/20	3.6/20
C_{expand}	3.6/22	3.6/21	3.6/20	3.6/20

time slack. Moreover, it gives significantly better hold time results for one of the testcases ($A*$).

4.7 Timing Aware Extensions

Though small wirelength has a good correlation with good timing, timing aware extensions to scan chain ordering are still desirable in many timing constrained designs. We may define, for example, the cost of a scan connection as *pin-to-net wirelength* $-\frac{x}{rc}\times$ (*average slack* + *minslack*) where $r(c)$ is resistance (capacitance) per unit length of interconnect. We will label this timing aware flow as Flow TAOur-TD.[6] Table IX reports timing results with $x = 0.25$. On average, the wirelength overhead for Flow TAOur-TD is twice as much as Flow PBQP-TD and five times more than Flow RBOur-TD. However, we do not see significant improvement in timing even at the cost of wirelength degradation.

We have outlined a more elaborate timing-driven scan chain reordering flow in Gupta et al. [2003]. We describe a method of finding the minimum wirelength incremental-connection point which meets all timingx requirements. We also describe a buffer insertion technique for connections which do not meet timing constraints. The flow was not validated because it requires the ability to route specific pin-to-tree connections. Modern backend data models and routers offer this capability via the *virtual-pin* ("vpin" in DEF) construct. On the other hand, the presence of many constraints appears to hamper traditional routing heuristics [Kahng and Mantik 2000]. The industry router that we use does not gracefully handle situations involving many such constraints, especially in its

[6]Since this flow requires timing information, it does not have the corresponding NTD case.

Table IX. Timing Aware Scan Chain Ordering Results
(Min Slack and Number of Setup Time Violations)

	Min Slack (ns)/# Setup Time Violations		
Test Case	PBQP-TD	RBOur-TD	TAOur-TD
A	5.24/602	5.50/578	5.12/632
A_{swap}	7.57/722	7.50/744	7.3/690
A_{expand}	6.02/817	6.26/781	6.35/776
B	5.25/4	5.26/4	5.26/6
B_{swap}	5.27/5	5.27/5	5.28/6
B_{expand}	5.23/4	5.23/4	5.24/4
C	15.63/5954	14.99/5725	14.5/5415
C_{swap}	24.38/6418	24.32/5973	23.5/5854
C_{expand}	26.38/7398	24.60/7320	24.9/7384

incremental routing mode, and the quality of the result appears to suffer even though the routing-based timing-driven scan ordering is better.

5. EXTENSIONS AND CONCLUSIONS

In this article, we have presented a new approach for routing-based scan chain ordering. Our main conclusions are as follows.

—A substantial reduction in wirelength impact of a scan is achieved by the routing-based flow compared with the traditional placement-based flow. The magnitude of this reduction ranges from 20.5% to 85.7% on industry test cases.

—There are timing benefits of routing-driven scan chain ordering. Typically, it has a positive impact on the hold time feasibility of the design.

—It is also clear that constraining the output pin to make the scan connection results in up to a 3x increase in the wirelength overhead of the scan.

While we have clearly demonstrated the positive impact of trial routing-based scan ordering, even better industry flows appear possible. For example, as we noted previously, the effectiveness of our approach may be limited by the capabilities of particular industry routing tools, for example, with respect to the quality of incremental optimization or the ability to follow virtual pin-based constraints. As another example, in congested layouts, a congestion-aware ordering capability may be required (based on both routing and congestion map information).

Finally, another interesting and practical extension of this work would be scan chain ordering with multiple scan chains. The case where the flip-flops in each scan chain are known gives a trivial extension (each chain is ordered independently). A more interesting problem is to simultaneously partition the scan flip-flops into multiple balanced scan chains.

REFERENCES

BARBAGELLO, S., BODONI, M. L., MEDINA, D., CORNO, F., PRINETTO, P., AND REORDA, M. S. 1996. Scan-insertion criteria for low design impact. In *Proceedings of VLSI Test Symposium*. 26–31.

BENTLEY, J. J. 1992. Fast algorithms for geometric traveling problems. *ORSA J. Comput. 4*, 4, 387–410.

BOESE, K. D., KAHNG, A. B., AND TSAY, R. S. 1994. Scan chain optimization: Heuristic and optimal solutions. Internal rep. UCLA Computer Science Dept. (Oct.). Available at http://www.gigascale.org/bookshelf/Slots/ScanOpt/.

FEUER, M. AND KOO, C. C. 1983. Method for rechaining shift register latches which contain more than one physical book. *IBM Tech. Disclos. Bull. 25*, 9, 4818–4820.

GUPTA, P., KAHNG, A. B., AND MANTIK, S. 2003. A proposal for routing-based timing-driven scan chain ordering. In *Proceedings of the IEEE International Symposium on Quality Electronic Design* (March). To appear.

HELSGAUN, K. 2000. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European J. Operat. Res. 12*, 106–130. The code is available at http://www.dat.ruc.dk/keld/research/LKH/.

HIRECH, M., BEAUSANG, J., AND GU, X. 1998. A new approach to scan chain reordering using physical design information. In *Proceedings of the International Test Conference*. 348–355.

HONG, I., KAHNG, A. B., AND MOON, B. R. 1997. Improved large-step markov chain variants for the symmetric TSP. *J. Heurist. 3*, 1, 63–81.

JOHNSON, D. S 1990. Local optimization and the traveling salesman problem. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*. 446–460.

JOHNSON, D. S. AND MCGEOCH, L. A. 1997. The traveling salesman problem: A case study in local optimization. In E. H. L. Aarts and J. K. Lenstra Eds. *Local Search Algorithms*, John Wiley and Sons, New York, NY.

JOHNSON, D. S., GUTIN, G., MCGEOCH, L. A., YEO, A., ZHANG, W., AND ZVEROVITCH, A. 2002. Experimental analysis of heuristics for the ATSP. *The Traveling Salesman and its Variations*. Kluwer Academic Publishers. To appear.

KAHNG, A. B. AND MANTIK, S. 2000. On mismatches between incremental optimizers and instance perturbations in physical design tools. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. (Nov.), 17–21.

KANELLAKIS, P. C. AND PAPADIMITRIOU, C. H. 1992. Local search for the asymmetric traveling salesman problem. *Operat. Res. Letters 11*, 219–224.

KOBAYASHI, S., EDAHIRO, M., AND KUBO, M. 1999. A VLSI scan-chain optimization algorithm for multiple scan-paths. *IEICE Transaction Fundamentals* E82-A(11), 2499–2504.

LAWLER, E. L., LENSTRA, J. K., RINNOOY-KAN, A., AND SHMOYS, D. 1985. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley and Sons, New York, NY.

LIN, K.-H., CHEN, C.-S., AND HWANG, T. T. 1996. Layout driven chaining of scan flip-flops. In *IEE Comput. Digit. Techn. 143*, 6, 421–425.

MAKAR, S. 1998. A layout based approach for ordering scan chain flip-flops. In *Proceedings of the International Test Conference*. 341–347.

MARTIN, O., OTTO, S. W., AND FELTEN, E. W. 1991. Large-step Markov chains for the traveling salesman problem. *Complex Syst. 5*, 3, 299–326.

MILLER, D. L. AND PEKNY, J. F. 1991. Exact solution of large asymmetric traveling salesman problems. *Science 251*, 15 (Feb.), 754–761.

REINELT, G. 1992. Fast heuristics for large geometric traveling salesman problems. *ORSA J. Comput. 4*, 2, 206–217.

ZHANG, W. 1992. On the expected complexity of the traveling salesman problem under subtour elimination. *UCLA Computer Science Dept. Tech. Rep. CSD-920022*.

Received March 2003; revised August 2004; accepted November 2004