Routing-Aware Scan Chain Ordering *

Puneet Gupta*, Andrew B. Kahng†† and Stefanus Mantik †
∗ ECE Department, University of California at San Diego
†† ECE and CSE Departments, University of California at San Diego
† Cadence Design Systems, Inc., San Jose, CA, USA
{puneet@ucsd.edu, abk@ucsd.edu and smantik@cadence.com}

Abstract

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 non-metric; 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 does also improve evaluated timing, and practical timing-driven extensions appear feasible.

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.

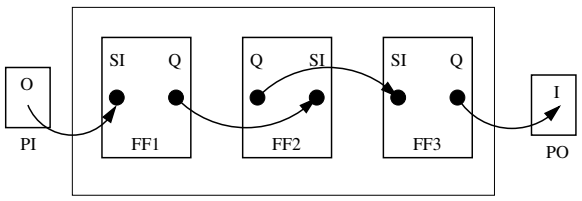

Figure 1: Example of a scan chain with three scan registers $FF1$, $FF2$, and $FF3$. In each sequential cell, *SI* and Q denote the scan-in pin and scan-out pin. *PI* is the primary input pad and *PO* is the primary output pad.

*Research at UCSD was supported by a grant from the MARCO Gigascale Silicon Research Center.

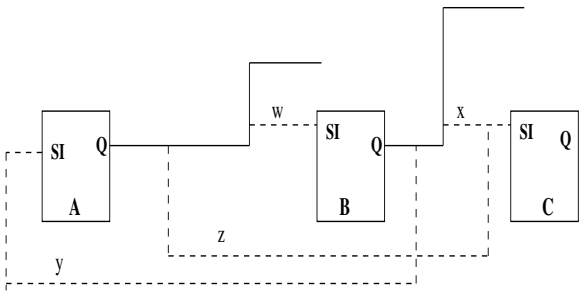

Figure 2: An example showing the highly asymmetric and non-metric nature of the ATSP when doing incremental scan-insertion.

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 increases wirability 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 [9], [19], [1] or pin-to-pin Manhattan distance [3], [16]. The former metric gives a symmetric TSP while the latter gives rise to an almost symmetric TSP [3]. 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 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 non-metric. We give an efficient method to calculate the costs of the ATSP instance, based on a trial routing of non-scan 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.
- This formulation of the TSP can be highly asymmetric. For instance, in Figure 2 $BA(=y) \gg AB(=w)$.
- We can also have non-metric 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 paper, 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 with final conclusions in Section 5.

2 Previous Work

In this section, we survey previous work in two relevant areas:

- Layout-based scan chain ordering approaches,
- 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. The scan orderings were essentially placement-based. The scan chain ordering problem is transformed to a symmetric or asymmetric TSP. Feuer and Koo [7] 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. [9], [19] and [1] 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. [3] and [16] modify 2-opt and 3-opt TSP heuristics to take into account the asymmetric nature of the scan chain problem. [18] 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. [17]. Later studies include a very comprehensive empirical comparison of symmetric TSP heuristics by Johnson [11]. Since Johnson's study, works by Bentley [2] and Reinelt [22] have suggested ways to reduce the running times of the more successful heuristics, while Martin et al. [20] 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 [15] which modifies the Lin-Kernighan (LK) symmetric TSP heuristic. More recent studies by Miller and Pekny [21] and Zhang [23] have applied branch-and-bound to obtain *optimal* solutions to asymmetric TSP. 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.

[10] 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. [20] used both LK and a fast implementation of 3-opt; [11] used LK only. Kick move perturbation of the current local minimum tour is typically achieved using a k'-change, with k' not necessarily equal to k. Both [20] and [11] use random "double-bridge" 4-change kick moves. According to [20], 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 [20] and the "iterated LK" heuristic of [11] are believed to be the best-performing iterated descent variants (and, indeed, the best-performing of all heuristics for obtaining near-optimal solutions [12]). [3] introduced restricted 2-opt and 3-opt moves to preserve directionality for solving an ATSP; their *ScanOpt*, an implementation of the LSMC heuristic with restricted moves for ATSP, is available at [4]. Finally, results of the *Eighth DIMACS Implementation Challenge* [6] for TSP are available in [13]. The method of [8] is shown to give the best tours if running time is not a constraint.

As suggested in Figure 2, asymmetry of the TSP will go up with use of any routing-based distance measure, while metricity decreases. We would like to assess the impact of the routing-based TSP distance metric on the nature of the resultant TSP instances. To this end, we adopt the asymmetry and metricity measures presented in [13]. 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]

[1] For 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 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 iff we get a value of 0. For non-negative distances, the above quantity has a maximum value of 1. We call this measure *non-metricity* since it increases with decreasing metricity. These measures are given in [13].

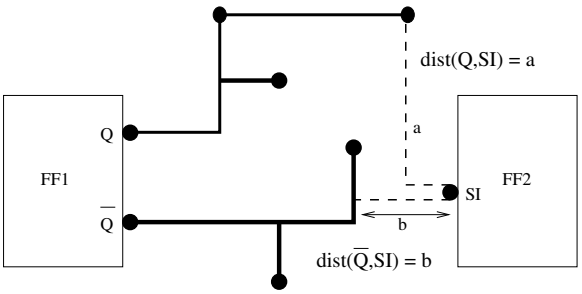

Figure 3: Pin-to-Net distance measurement for routing-driven scan ordering.

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 future extension, at the end of this paper.

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 scan-out pin (Q or $\bar{Q}$) to the next 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). $dist(Q,SI)$ (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 to obtain 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*, i.e., the routing starts from an unrouted instance. Note that our flow is not an ECO flow.[2] [14] demonstrates that incremental routing can achieve lower solution quality than from-scratch routing, and that the magnitude of change in the instance can strongly affect the resulting quality of result. Thus, we use the from-scratch routing for better solution despite of the high 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 choice of TSP solver, and on 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 v5.3.125* (SE) and Cadence *QPlace v5.1.68* as the physical design tool to perform the industry placement-based scan chain ordering. In addition, we use Cadence *WRoute v2.2.31* 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 below.

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. `Non-Timing Driven (NTD):` When the timing library is not supplied, QPlace will optimize the placement based on wirelength (and possibly congestion).
   - B. `Timing Driven (TD):` By supplying the timing library and timing constraints, QPlace will try to optimize the timing (i.e., minimizing the incidence of negative slacks).
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 the existing net of the scan-out pin and by adding a new net for the scan-out pin that does not have any previous connection (the scan-out scan-in pair follows the order specified by the ATSP solver).

[2] ECO (Engineering Change Order) is an incremental type of flow that uses a pre-routed 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.

3. `WRoute:` The placed netlist is routed using *WRoute*. Similarly to the variations in placement, there are also two variations in the routing.

   A. `Non-Timing 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, LKH):` 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 above-mentioned steps are used to construct the following scan chain insertion flows.

- *Flow Ia: 1A,3A.* Baseline NTD total wirelength with no scan nets.
- *Flow Ib: 1B,3B.* Baseline TD total wirelength and timing with no scan nets.
- *Flow IIa: 1A, 2A, 3A.* NTD placement-based SE flow.
- *Flow IIb: 1B, 2A, 3B.* TD placement-based SE flow.
- *Flow IIIa: 1A, 2B, 3A.* NTD placement-based QP-Scan flow.
- *Flow IIIb: 1B, 2B, 3B.* TD placement-based QP-Scan flow.
- *Flow IVa: 1A, 2C, 3A.* Our NTD placement-based flow. This directly compares with Flow IIa and Flow IIIa, i.e., 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 IVb: 1B, 2C, 3B.* Our TD placement-based flow. Similarly, this directly compares with Flow IIb and Flow IIIb.
- *Flow Va: 1A, 3A, 4A, 3A.* NTD routing-based SE flow.
- *Flow Vb: 1B, 3B, 4A, 3B.* TD routing-based SE flow.
- *Flow VIa: 1A, 3A, 4B, 3A.* NTD routing-based QP-Scan flow.
- *Flow VIb: 1B, 3B, 4B, 3B.* TD routing-based QP-Scan flow.
- *Flow VIIa: 1A, 3A, 4C, 3A.* Our NTD routing-based flow. This directly compares with Flow Va and Flow VIa, i.e., industry vs. our routing-based scan ordering solvers. Again, the comparison is clouded by possible differences in costing and objective function. Flow VIIa also compares to Flow IVa to assess the impact of routing-based ordering vis-a-vis placement based ordering.
- *Flow VIIb: 1B, 3B, 4C, 3B.* Our TD routing-based flow. Similarly, this directly compares with Flow Vb and Flow VIb.

We extracted scan chain orders from the industry tool's flow. Scan chain orders generated by *SE* in Flow IIa(b) and Flow Va(b) are identical. Similarly, scan chain orders generated by *QPlace* in Flow IIIa(b) and Flow VIa(b) are identical. *We interpret this as the absence of any routing-based ordering in both SE and QPlace*. Therefore, Flows Va, Vb, VIa, and VIb become redundant and we do not report results for them. In total, there are ten distinct results for each test case, corresponding to Flows Ia, Ib, IIa, IIb, IIIa, IIIb, IVa, IVb, VIIa, and VIIb.

4.2 ATSP Solver Comparison

As our results heavily depend on the quality of the TSP tour, we compare two ATSP solvers *ScanOpt* [4] and *LKH-1.2* [8]. *LKH-1.2* is reported to be one of the best ATSP solvers currently available [13]. *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

	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

Table 1: A comparison of ATSP solvers.

of our test cases (i.e., *A* with 1226 cities) is given in Table 1. With two different measures (tour cost and CPU time), on test case A, *ScanOpt* is more favorable with respect to running time while *LKH* may give a better tour with some extra runtimes.[3]

4.3 Test Cases

We consider three test case obtained from industry sources. Each of the test cases was obtained in LEF/DEF format and then modified to merge its multiple scan chains into one scan chain. We then generated alternate placements for each test case by (1) relaxing the site map by 20% and (2) randomly swapping some scan flip-flop locations.[4] Main parameters of the test cases are given in Table 2. A_{swap} denotes the test case with placement of *A* altered by randomly swapping the placements of scan flip-flops. A_{expand} denotes the test case obtained by increasing the site map of *A*. The asymmetry and non-metricity values for the corresponding TSP instances are shown in Table 3. Note that non-metricity 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 Results

Table 4 shows the total wirelength after detailed routing for the baseline Flow I and scan overhead for each of the remaining four flows

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

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

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 2: Characteristics of the test cases.

Non-TD Instance	Asymmetry/Non-Metricity		
	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

Table 3: Asymmetry and (Non-)Metricity measures for the test cases.

explained in the 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 5 (CPU times are normalized to a 143Mhz SUN Ultra-1). CPU time for Flow VIIa and VIIb are the sum of the initial routing time (before scan-insertion) and the final routing time (after the scan-insertion). Table 6 shows the effect on timing violations and slacks. We use setup violations for the timing measurements.

Table 4 clearly shows that both Flow VIIa and VIIb produce better wirelength than Flow IVa and Flow IVb respectively. This gives an unbiased comparison of the *placement-based* ordering and the *routing-based* ordering. Flows VIIa and VIIb also have better wirelength than the industry flows, Flow IIa, IIb, IIIa, and IIIb. The reduction in wirelength ranges from 20.5% to 85.7%. This shows that by exploiting extra information we can indeed achieve better solution. Furthermore, *QP*-Scan (Flow IIIa(b)) clearly has better TSP solver than *SE* (Flow IIa(b)) while our ATSP solver (Flow IVa(b)) is as good as *QP*-Scan. There are three cases where Flow VIIa(b)

Test Case (NTD)	Total WL (μm)	Scan Overhead(μm)			
	Flow Ia	Flow IIa	Flow IIIa	Flow IVa	Flow VIIa
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
(TD)	Flow Ib	Flow IIb	Flow IIIb	Flow IVb	Flow VIIb
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

Table 4: Post-detailed routing wirelength and scan overhead for various scan chain insertion flows ($*$ indicates that the routing completed with violations).

completes the routing with violations; however, with these cases, the industry flow has violations as well.

In the timing domain (Table 6), 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 magnitudes of the timing violations (i.e., negative slacks) are not worse than the industry flows in some cases.

5 Conclusions

In this paper, 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 scan is achieved by the routing-based flow, as compared with the traditional placement-based flow. The magnitude of this reduction ranges from 20.5% to 85.7% on industry test cases.
- Although our current experiments are timing-oblivious, the routing-based flow reduces the number of timing violations while the magnitudes of the timing violations do not degrade significantly from the industry flow.

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 above, the effectiveness of our approach may be limited by the capabilities of particular industry routing tools, e.g., with respect to quality of incremental optimization or 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).

The focus of our ongoing and future work is on extensions to true timing-driven scan ordering. Adding scan connections can cause ad-

Test Case	Flow Ia	Flow IIa	Flow IIIa	Flow IVa	Flow VIIa
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	Flow Ib	Flow IIb	Flow IIIb	Flow IVb	Flow VIIb
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 5: Total router CPU times (in seconds) for various scan chain insertion flows normalized to SUN Ultra-1 at 143MHz. CPU time for Flow VIIa and VIIb are the sum of the initial routing time (before scan-insertion) and the final routing time.

	Min Slack (ns)/# Timing Violations			
Test Case	Flow IIb	Flow IIIb	Flow IVb	Flow VIIb
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

Table 6: Timing results for the TD scan-insertion flows.

ditional timing violations. We are developing a (detailed) routing-based scan ordering flow that can compute the route with the least wirelength impact that attaches the *SI* pin of ff_{in} to the output of ff_{out}, so as to maximize timing slack with respect to timing constraints on any of the sinks of the fanout tree of ff_{out}. An enabling observation is that the optimum attachment point on any given segment of the fanout tree of ff_{out} can, under the Elmore delay model, be found as the solution to a simple analytic equation. (Even brute-force search over all tree segments for the best attachment point is computationally inexpensive, and speedups are possible.) The optimum attachment point can also be efficiently found under the assumption that buffering of the connection (to meet timing constraints) is a degree of freedom for the scan ordering step. The regime where only a limited number of buffer sites are available (and, are competed for by different potential scan connections) offers interesting challenges within the ATSP framework. 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 is 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

[1] S. Barbagello, M.L. Bodoni, D. Medina, F. Corno, P. Prinetto and M.S. Reorda, "scan-insertion Criteria for Low Design Impact", *Proc. VLSI Test Symposium*, 1996, pp. 26-31.

[2] J.J. Bentley, "Fast Algorithms for Geometric Traveling Problems", *ORSA Journal of Computing*, 4(4) (1992), pp. 387-410.

[3] K.D. Boese, A.B. Kahng and R.S. Tsay, "Scan Chain Optimization: Heuristic and Optimal Solutions", Internal Report, UCLA CS Dept., October 1994. Downloadable from *http://www.gigascale.org/bookshelf/Slots/ScanOpt/*

[4] GSRC Bookshelf, "Scan Chain Optimization", *http://www.gigascale.org/bookshelf/Slots/ScanOpt/*

[5] C.S. Chen and T.T. Hwang, "Layout Driven Selection and Chaining of Partial Scan Flip-Flops", *Journal of Electronic Testing: Theory and Applications* 13 (1998), pp. 19-27.

[6] "Eighth DIMACS Implementation Challenge", *http://www.research.att.com/ dsj/chtsp/*, 2002.

[7] M. Feuer and C. C. Koo, "Method for Rechaining Shift Register Latches Which Contain More Than One Physical Book", *IBM Technical Disclosure Bulletin* 25 (9) (1983), pp. 4818-4820.

[8] K. Helsgaun, "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic", *European Journal of Operations Research* 12 (2000), pp. 106-130. The code is available at *http://www.dat.ruc.dk/ keld/research/LKH/*.

[9] M. Hirech, J. Beausang and X. Gu, "A New Approach to Scan Chain Reordering Using Physical Design Information", *Proc. International Test Conference*, 1998, pp. 348-355.

[10] I. Hong, A.B. Kahng and B.R. Moon, "Improved Large-Step Markov Chain Variants for the Symmetric TSP", *Journal of Heuristics* 3(1) (1997), pp. 63-81.

[11] D.S. Johnson, "Local Optimization and the Traveling Salesman Problem", *Proc. 17th Intl. Colloquium on Automata, Languages and Programming*, 1990, pp. 446-460.

[12] D.S. Johnson and L.A. McGeoch, "The Traveling Salesman Problem: A Case Study in Local Optimization", In E.H.L. Aarts and J.K. Lenstra, editors, *Local Search Algorithms*, Wiley and Sons, New York, 1997.

[13] D.S. Johnson, G. Gutin, L.A. McGeoch, A. Yeo, W. Zhang and A Zverovitch, "Experimental Analysis of Heuristics for the ATSP", To appear in *The Traveling Salesman and its Variations*, Kluwer Academic Publishers, 2002.

[14] A. B. Kahng and S. Mantik, "On Mismatches Between Incremental Optimizers and Instance Perturbations in Physical Design Tools", *Proc. IEEE/ACM Intl. Conference on Computer-Aided Design*, November 2000, pp. 17-21.

[15] P.C. Kanellakis and C.H. Papadimitriou, "Local Search for the Asymmetric Traveling Salesman Problem", *Operations Research Letters* 11 (1992), pp. 219-224.

[16] S. Kobayashi, M. Edahiro and M. Kubo, "A VLSI Scan-Chain Optimization Algorithm for Multiple Scan-Paths", *IEICE Trans. Fundamentals* E82-A(11) (1999), pp. 2499-2504.

[17] E.L. Lawler, J.K. Lenstra, A. Rinnooy-Kan and D. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, 1985.

[18] K.-H. Lin, C.-S. Chen and T.T. Hwang, "Layout Driven Chaining of Scan Flip-flops", *IEE Proc., Computers and Digital Techniques* 143(6) (1996), pp. 421-425.

[19] S. Makar, "A Layout Based Approach for Ordering Scan Chain Flip-Flops", *Proc. International Test Conference*, 1998, pp. 341-347.

[20] O. Martin, S.W. Otto and E.W. Felten, "Large-step Markov Chains for the Traveling Salesman Problem", *Complex Systems* 5(3) (1991), pp. 299-326.

[21] D.L. Miller and J.F. Pekny, "Exact Solution of Large Asymmetric Traveling Salesman Problems", *Science* 251, 15 February 1991. pp. 754-761.

[22] G. Reinelt, "Fast Heuristics for Large Geometric Traveling Salesman Problems", *ORSA Journal on Computing*, 4(2) (1992), pp. 206-217.

[23] W. Zhang, "On the Expected Complexity of the Traveling Salesman Problem under Subtour Elimination", *UCLA CS Dept. Tech. Report CSD-920022*, 1992.