Incremental Multiple-Scan Chain Ordering for ECO Flip-Flop Insertion

Andrew B. Kahng[†‡], Ilgweon Kang[‡], and Siddhartha Nath[‡]
UC San Diego ECE[†] and CSE[‡] Departments, La Jolla, CA 92093
{abk, igkang, sinath}@ucsd.edu

ABSTRACT

Testability of ECO logic is currently a significant bottleneck in the SOC implementation flow. Front-end designers sometimes require large functional ECOs close to scheduled tapeout dates or for later design revisions. To avoid loss of test coverage, ECO flip-flops must be added into existing scan chains with minimal increase to test time and minimal impact on existing routing and timing slack. We address a new Incremental Multiple-Scan Chain Ordering problem formulation to automate the tedious and time-consuming process of scan stitching for large functional ECOs. We present a heuristic with clustering, incremental clustering and ordering steps to minimize the maximum chain length (test time), routing congestion, and disturbance to existing scan chains. Test times for our incremental scan chain solutions are reduced by 5.3%, and incremental wirelength costs are reduced by 45.71%, compared to manually-solved industrial testcases.

1. INTRODUCTION

In the design of semiconductor chips, scan is one of the most popular techniques for design for testability (DFT). Designers must achieve an adequate level of test coverage for the circuit [1]. Sometimes, when chips are very close to tapeout, front-end designers may require sudden modifications adding hundreds of flip-flops (FFs) and thousands of logic gates in total. To avoid loss of test coverage, the incremental scan FFs should be included into existing scan chains during the implementation of such engineering change orders (ECOs). This yields an *incremental multiple-scan chain ordering* problem, for which key considerations are as follows.

- The new ECO should be distributed among the existing scan chains so as to minimize test time, which is very expensive and is determined by the maximum chain length.
- Depending on the clock domain of each added ECO FF, only a subset of the existing scan chains will be able to accommodate, i.e., are *compatible with*, that FF.
- As a completed design may already contain routing congestion, it is desirable to stitch the ECO FFs into existing scan chains so as to minimize perturbations to existing routing, and hence minimize impact on timing.

The resulting optimization trades off test time against routability and total wirelength. Figure 1 illustrates the incremental multiple-scan chain ordering problem. The red cells represent incremental scan flip-flops (ECO FFs) that have been stitched appropriately into nearby compatible scan chains which had existed previously in the design. Late-stage manual scan stitching can cost days or weeks of design time, whereas a suitable tool could decrease this schedule hit to a day or less.

In this work, we formulate the *incremental multiple-scan chain ordering* (IMSCO) problem, and propose a heuristic method to add ECO FFs into existing scan chains in the original design. We perform assignments of ECO FFs to scan chains by cluster analysis. Then, we incrementally order each group of assigned ECO FFs within its respective scan chain with small added wirelength using traveling salesman problem (TSP) heuristics.

Our main contributions are the following.

- We add ECO FFs to existing scan chains in the original design while satisfying a given maximum scan chain depth constraint.
- We heuristically minimize total incremental routing length (optionally, weighted according to a given congestion map), and we also try to minimize the number of cut edges in original chains since these can cause unexpected side effects to the original design.
- To our knowledge, we are the first to address the incremental multiple-scan chain ordering problem at the ECO level. We provide an efficient approach that meets practical demands of physical designers.

Of independent interest in the operations research and metaheuristics contexts is that our incremental multiple-scan chain ordering problem can be seen as a new *dynamic* multi-depot vehicle routing problem with a variable number of movable depots.

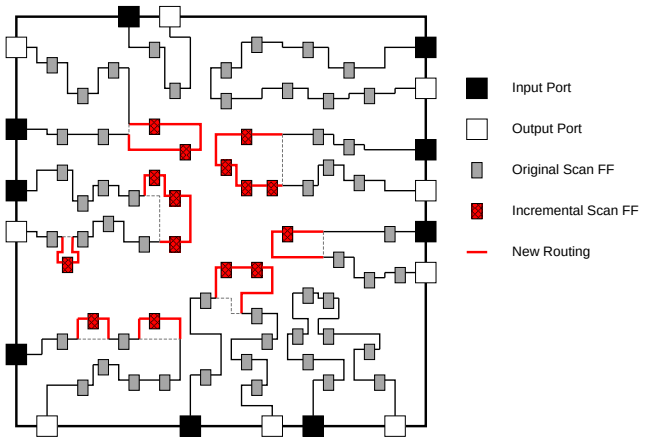


Figure 1: Incremental ordering of multiple scan chains when flip-flops are inserted into the design.

The remainder of this paper is organized as follows. In Section 2, we review related works. Section 3 gives notation and definitions, and formally states the incremental scan chain clustering and ordering problems that we address. Section 4 presents our heuristics, and Section 5 describes experimental setup, results, and discussion. Section 6 provides directions for ongoing work and concludes the paper.

2. RELATED WORKS

IMSCO consists of two subproblems, (1) clustering and assignment of ECO FFs into original scan chains, and (2) ordering of the assigned ECO FFs. Clustering, assignment and stitching of ECO FFs at cut edges in original scan chains can be seen as a variant of the multi-depot vehicle routing problem (MDVRP), in which one vehicle can be assigned to each depot (i.e., the depot is the start/end location of the scan chain, and the scan order is

IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2013, November 18-21, 2013, San Jose, California, USA

the vehicle's route). The vehicle routing problem (VRP) [5] seeks optimal delivery or collection routes from one depot to a number of geographically scattered cities (or customers), subject to side constraints [17]. MDVRP is a variant of VRP that considers multiple depots at the same time. Both VRP and MDVRP are well-known to be NP-hard. Ordering of a group of ECO FFs within a previously-existing scan chain can be formulated as an incremental traveling salesman problem (TSP). The TSP, too, is a well-known NP-hard combinatorial optimization problem [10] and has been extensively studied in various fields.

Application of clustering to scan chains. Elm et al. [7] present linear-time partitioning heuristics to cluster scan FFs into scan chains to reduce test power. They do not apply any ordering refinements, hence their method can potentially cause severe performance variations during test. Seok et al. [25] reduce test time by dividing a scan chain into multiple scan chains based on placement information. Then they reorder scan FFs based on test vectors. In addition, their results show improvement of congestion. However, they do not formulate the scan chain optimization as an ECO problem.

Application of VRP/MDVRP for scan chain assignment. The first exact algorithm for MDVRP is due to Laporte et al. [16], but is useful only for small instance sizes. Renaud et al. [24] and Cordeau et al. [6] present Tabu search-based MDVRP heuristics. The instance sizes handled by these works are too small for application to ECO scan chain optimization. Pepin et al. [20] and Gendreau et al. [11] survey several heuristic approaches for VRP and MDVRP. Ho et al. [14] and Mirabi et al. [19] propose other heuristics but again consider limited numbers of customers and depots which are small compared to IMSCO instances. The multiple traveling salesman problem (mTSP) is a generalization of TSP with more than one salesman and is an alternative to VRP [4]. However, prior work rarely considers salesmen from multiple depots in mTSP, i.e., each salesman must begin and end at a single prescribed depot.

Application of TSP for scan chain ordering. Lawler et al. [18] and Johnson [15] comprehensively review and empirically compare TSP heuristics. Johnson provides tour construction heuristics such as nearest-neighbor, as well as local optimization heuristics such as 3-Opt, Lin-Kernighan (LK) and iterated LK. Feuer and Koo [8] present the first published work on application of TSP for scan chain optimization by translating a given scan chain instance into a TSP instance. However, they lose geometric information in this translation, and hence cannot quantify the effectiveness of the TSP-based approach.

Gupta et al. [12] propose a routing-driven methodology for scan chain ordering that uses incremental routing cost (connecting to existing or anticipated routing, rather than to the output pin) as the cost for a scan connection. They consider scan chain ordering as an asymmetric TSP, and use *ScanOpt* [27] based on 3-Opt as their TSP solver. However, [12] does not consider timing; a subsequent method in [13] is both routing-driven and timing-aware. They search for the minimum wirelength incremental connection that meets timing constraints, and perform buffer insertion for connections that do not meet timing constraints. While their methods are quite reasonable, they are not useful for ECO scan FF ordering because they do not consider constraints such as scan time, number of cut edges, etc. that are important during ECO scan FF insertion. Rahimi et al. [21] assign scan FFs to multiple scan chains using partitioning and multichain assignment instead of TSP. Raul et al. [22] assign and order multiple scan FFs by using physical layout information. These methods are directed only to initial constructions of scan chains and not to the ECO context.

3. PROBLEM FORMULATION

Table 1 defines notations pertaining to our IMSCO problem formulation. We also use the following terminology.

Incremental Scan FF (ECO FF). Incremental scan FFs are scan FFs added to the design during a functional ECO, occurring when the design has reached near-final stages of physical implementation and a restart from synthesis is too costly. *Each incremental scan FF must be added to a compatible scan chain.*

Compatible Scan Chain. A scan chain is compatible with a given ECO FF if the scan chain contains FFs in the same clock domain as the ECO FF.

Test Time. Test time is proportional to the maximum length of any scan chain, assuming that all scan shifts are executed in parallel. *To minimize test time, the maximum scan chain length should be minimized.*

Incremental Wirelength. ECO routing is required to stitch ECO FFs into original scan chains. *To minimize the number of buffers added to scan chains and prevent high congestion, total wirelength for ECO routing should be minimized.*

Cut Edge. Cut edges are nets on original scan chains that are cut when ECO FFs are inserted. In general, *the number of cut edges should be minimized to reduce disturbance to the existing routing, since changes to existing routes may impact previously-achieved timing closure.*

Table 1: Notations used in our IMSCO formulation.

Term	Meaning
C	Set of scan chains, $\{c_k\}$, $c_k \in C$, $1 \leq k \leq \|C\|$
F_O	Set of original scan FFs
F_E	Set of incremental scan FFs (ECO FFs)
F	Set of scan FFs, $\{F_O\} \cup \{F_E\}$
F_U	Set of unwired ECO FFs, $F_U \subseteq F_E$
F_k	Set of scan FFs in chain c_k
f_i	i^{th} scan FF, $1 \leq i \leq \|F\|$
C_{C_i}	Set of scan chains that are compatible with f_i, $C_{C_i} \subseteq C$
$d(i,j)$	(Manhattan) distance between scan FFs f_i and f_j
$p(i,j)$	0-1 variable to indicate if f_i and f_j are connected
$g(i,k)$	0-1 variable to indicate if f_i is assigned to c_k
D_{MAX}	Upper bound on scan chain depth
L_{MAX}	Upper bound on distance between connected f_i and f_j
L_{MIN}	Lower bound on distance between connected f_i and f_j
E_{MAX}	Upper bound on the number of cut edges in the design
e_{MAX}	Upper bound on the number of cut edges in the chain
Constraints	$\{D_{MAX}, L_{MAX}, L_{MIN}, E_{MAX}, e_{MAX}\}$

Three optimization problems for ECO FF insertion. We have studied three variants of ECO FF insertion.

MinTT-ISC (minimize test time). In this problem, we insert ECO FFs into compatible scan chains such that the length (scan depth) of each scan chain is less than an upper bound D_{MAX}. The problem is formally stated as:

$$\text{minimize} \max_{c_k \in C_{C_i}} \sum_{f_i \in F} g(i,\ k)$$

$$\text{subject to} \sum_{\substack{f_i \in F \\ c_k \in C_{C_i}}} g(i,\ k) \leq D_{MAX}$$

MinIncWL-ISC (minimize incremental wirelength). It is desirable to stitch ECO FFs into the original scan chains with minimum possible wirelength. By minimizing wirelength, design engineers can also minimize the number of buffers required to close timing and meet design rule constraints. Minimal increase in wirelength also implies minimal disturbance to existing routing, which can help with functional mode timing closure. Our formulation also considers a lower bound on distance to avoid introducing hold time violations in scan shift mode. We formulate this problem as:

$$\text{minimize} \sum_{f_i, f_j \in F_E} d(i,\ j) \cdot p(i,\ j) + \sum_{\substack{f_i \in F_O \\ f_j \in F_E}} d(i,\ j) \cdot p(i,\ j)$$

$$\text{subject to } L_{MIN} \leq d(i,\ j) \cdot p(i,\ j) \leq L_{MAX}$$

where $p(i,j) = 1$ if f_i and f_j are connected, 0 otherwise.

MinCE-ISC (minimize number of cut edges). To minimize the impact of re-routing during ECO implementation, we can also minimize the number of cut edges in the original scan chains. We formulate this problem as:

$$\text{minimize} \sum_{\substack{f_i \in F_O \\ f_j \in F_E}} p(i,\ j)$$

$$\text{subject to} \sum_{\substack{f_i \in F_O \\ f_j \in F_E}} p(i,\ j) \leq E_{MAX}$$

$$\sum_{\substack{f_i \in F_k \\ f_j \in F_E}} p(i,\ j) \leq e_{MAX}$$

4. OPTIMIZATION OF ECO FF INSERTION

We now describe our heuristic approach for the IMSCO problem. We propose a three-phase heuristic with (1) initial greedy clustering of ECO FFs based on an *affinity vector*; (2) iterative improvement of initial clustering using a pass-based, Fiduccia-Mattheyses (FM) style algorithm [9] based on movement of single ECO FFs; and (3) multiple cut edge selection in original scan chains based on k-way clustering of ECO FFs. Our approach heuristically addresses the goals of minimum scan chain depth[1], minimum wirelength, and minimum number of cut edges. Figure 2 shows the flow of our proposed approach.

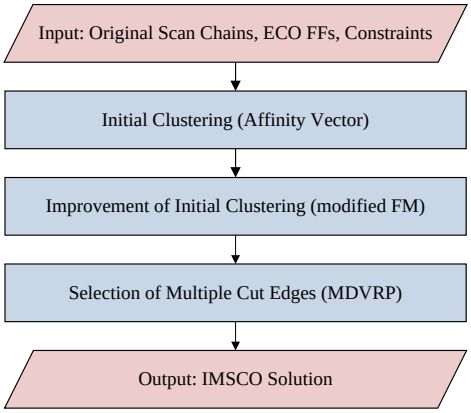


Figure 2: Our IMSCO flow.

4.1 Initial Clustering

Algorithm 1 describes the initial clustering phase. We use an *affinity vector function* that considers three objectives: (1) scan chain depth (SD), (2) wirelength (WL), and (3) cut edges (CE). To construct the initial cluster that simultaneously optimizes SD, WL, and CE, we formulate the affinity vector function as a weighted sum of each normalized affinity vector of SD, WL and CE. For each of SD, WL and CE, we calculate all possible connections between original scan FFs and ECO FFs, and between assigned ECO FFs and unassigned ECO FFs. For CE, we seek to minimize the number of cut edges but do not select the specific edges to cut until a "selection of multiple cut edges" phase, described in Section 4.3 below.

$Affinity(f_i)$ is a vector of size $|C|$ of affinity values $\langle Affinity(f_i)_1, Affinity(f_i)_2, \ldots, Affinity(f_i)_{|C|}\rangle$ of a given assigned ECO FF f_i to each scan chain.

$$Affinity(f_i)_k = \alpha \cdot SD(i,k) + \beta \cdot WL(i,k) + \gamma \cdot CE(i,k) \quad (1)$$

where $f_i \in F_{unassigned}$, $c_k \in C$, and $\{\alpha, \beta, \gamma\}$ are empirically determined. $SD(i,k)$, $WL(i,k)$ and $CE(i,k)$ in Algorithm 1 respectively compute SD, WL and CE affinities. To compute SD affinity, we define $SD(c_k)$ as the current scan depth of scan chain c_k and calculate the *SD_limit* across all scan chains as

$$SD\_limit = \min\{\max_{c_k}\{SD(c_k)\}, D_{MAX}\} \quad (2)$$

SD_limit contrains the assignment of ECO FFs to scan chains such that SD values across all scan chains remain balanced. When an ECO FF is assigned to a scan chain c_k, the value of $SD(c_k)$ increases by one. To promote assignment of ECO FFs to scan chains with smaller $SD(c_k)$, the SD affinity function returns the reciprocal of $SD(c_k)$. When an ECO FF is assigned to a scan chain c_k with $SD(c_k) < SD\_limit$, the value of *SD_limit* remains unchanged. However, when an ECO FF is assigned to a scan chain c_k with $SD(c_k) = SD\_limit$, the value of *SD_limit* increases by one as long as the new value is $\leq D_{MAX}$. Figure 3 illustrates these two cases of ECO FF assignment.

To compute WL affinity, we use the reciprocal of the distance between the ECO FF f_i and a nearest FF in the scan chain c_k. The distance functions for congestion- and non-congestion-aware modes are different. In non-congestion-aware mode the distance between two scan FFs is the Manhattan distance, whereas in congestion-aware mode the distance is calculated as

$$Distance(f_i,\ f_j) = x_{Con} \cdot |x_i - x_j| + y_{Con} \cdot |y_i - y_j| \quad (3)$$

where x_{Con} and y_{Con} are congestion coefficients in the horizontal and vertical directions, calculated as the average ratio of used tracks to the total number of tracks for all grid cells in the bounding box $\{(x_i,\ y_i),\ (x_j,\ y_j)\}$.

To compute CE affinity, we choose an edge in a scan chain c_k that is nearest to the ECO FF f_i. We then use the reciprocal of the number of "virtual" cut edges as shown in Figure 4.

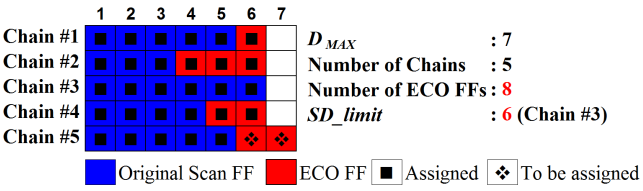


Figure 3: Example of ECO FF assignment and impact on SD. In the figure, there are eight ECO FFs out of which six have been assigned to scan chains #1–#5. **Case 1.** The seventh ECO FF is assigned to Chain #5. SD of Chain #5 increases to six, but *SD_limit* remains at six. **Case 2.** The eighth ECO FF is assigned to Chain #5. *SD_limit* must change to seven because all scan chain depths are six after the seventh ECO FF is assigned to Chain #5.

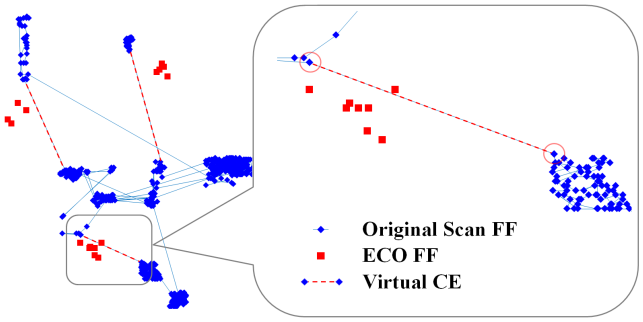


Figure 4: Virtual cut edges in the CE affinity calculation. There are three virtual CEs in the figure.

Algorithm 1 invokes the *InitialClustering* procedure with the set of original scan chains, ECO scan FFs and constraints. Initially, all ECO FFs are put into an $F_{unassigned}$ list. We pick an ECO FF, f_i, from this list and invoke the *Affinity* procedure to calculate the affinity of f_i to all compatible scan chains C_{C_i}. If some constraint

[1] We use scan chain depth to refer to scan chain length.

is not satisfied for a scan chain, then the affinity value of f_i for the scan chain is set to -1. If all affinity values are negative, then f_i is moved to a set of unwired ECO FFs. In each iteration we decrease the number of elements in $F_{unassigned}$ by one and continue until all ECO FFs are assigned to some scan chain or are added to the set of unwired ECO FFs. We calculate the bias for each ECO FF as the difference between the two largest affinity values across all scan chains and assign it to the scan chain for which it has the largest value of affinity. We then invoke the TSP solver *Concorde* [2] to find the optimal tour of all the assigned ECO FFs in each scan chain. (1) Thus, at the end of *InitialClustering* we have clustered a list of ECO FFs for each scan chain and ordered them using a TSP heuristic to minimize wirelength. (2) Details of our ordering scheme are given below in Section 4.4.

Algorithm 1 Procedures to assign ECO scan flip-flops.

Procedure *InitialClustering*(F, C, *Constraints*)
Input: scan chains, ECO FFs, constraints
Output: clustering and ordering of ECO FFs; assignment of ECO FFs to original scan chains
$F_{unassigned} \leftarrow F_E$;
while $F_{unassigned} \neq \phi$ **do**
  $f_i \leftarrow F_{unassigned}.pop$;
  $F_{unassigned}.size \leftarrow F_{unassigned}.size - 1$;
  Affinity(f_i) $\leftarrow$ *Affinity*(f_i, C, *Constraints*);
end while
while $i \leq F_E.size$ **do**
  $Bias \leftarrow Affinity(f_i)_{|C|} - Affinity(f_i)_{|C|-1}$;
  if $Bias == 0$ **then**
    $F_U.push(f_i)$;
  end if
  $i \leftarrow i+1$;
end while
Concorde(C, $F_E \setminus F_U$); // TSP solver

Procedure *Affinity*(f_i, C, *Constraints*)
for each $c_k \in C_{C_i}$ **do**
  if f_i does not satisfy some *Constraints* **then**
    $Affinity(f_i)_k \leftarrow -1$;
  else
    $Affinity(f_i)_k \leftarrow \alpha \cdot SD(i,k) + \beta \cdot WL(i,k) + \gamma \cdot CE(i,k)$;
  end if
  $Affinity(f_i).push(Affinity(f_i)_k)$;
end for
$Affinity(f_i).sort$;
return *Affinity*(f_i);

Procedure $SD(i,k)$
if $SD(c_k) \leq SD\_limit$ **then**
  $SD(c_k) \leftarrow SD(c_k) + 1$;
  return $\{1 / SD(c_k)\}$;
else
  return $\{-1\}$;
end if

Procedure $WL(i,k)$
distance(f_i, c_k) $\leftarrow$ distance between f_i and nearest f_j in c_k;
return $\{1 /$ distance(f_i, c_k)$\}$;

Procedure $CE(i,k)$
virtualCE $\leftarrow$ number of virtual CEs after assigning f_i to c_k;
return $\{1 /$ virtualCE$\}$;

4.2 Improvement of Initial Clusters

We create initial clusters by applying a greedy method based on the value of the affinity vector. Usually, greedy heuristics become easily stuck in local minima. To improve the initial solution, we perform iterative improvement using a modified FM strategy, based on single moves of ECO FFs. From the given initial clustering solution, we consider all possible single moves of ECO FFs to another scan chain. Since we already minimize the scan chain depth during initial clustering, we focus on reducing incremental WL in the iterative improvement procedure. Equation (4) gives the gain function we use in our iterative improvement procedure.

$$GainWL(F_E, C) = WL(\text{removed edges}) - WL(\text{added edges}) \quad (4)$$

A single move of an ECO FF from scan chain c_j to scan chain c_i removes three edges (two edges from c_j and one edge from c_i) and adds three edges (one edge from c_j and two edges from c_i). For all combinations of ECO FFs and scan chains, we search for the ECO FF f_i that gives maximum gain, and move f_i from c_j to c_i. Note that a move is performed only when the depth of scan chain c_i is $< D_{MAX}$. After being moved, the ECO FF is fixed in the scan chain c_i. When all ECO FFs have been moved once, this completes one pass of iterative hill-climbing improvement. If the iterative improvement reduces total WL, we run another iterative improvement step. This continues until there is no more improvement in WL.

To reduce the number of hill-climbing iterations, we use two termination conditions: (1) *If {height of hill $>$ 7% of temporary minimum WL}, then exit*; and (2) *If {local minimum && #iterations $>$ one quarter of the number of ECO FFs}, then exit*. We determine these conditions empirically. Algorithm 2 presents our iterative improvement procedure.

Algorithm 2 Improvement of Initial Solution.

Procedure *Improvement*(F, C, *Constraints*)
Input: scan chains, ECO FFs, constraints
Output: improvement of initial solution
execute the modified FM($GainWL(F_E, C)$);
Concorde(C, $F_E \setminus F_U$); // TSP solver

Algorithm 3 Selection of Multiple Cut Edges.

Procedure *MDVRP*(F, C, *Constraints*)
Input: scan chains, ECO FFs, constraints
Output: scan FF ordering with multiple cut edges
while the number of CEs in the design $< E_{MAX}$ **do**
  for $\forall c_k$ with #CEs $< e_{MAX}$ **do**
    calculate *gain* $\forall$ clusters in c_k;
  end for
  find $gain_{MAX}$;
  if $gain_{MAX} > 0$ **then**
    divide cluster into two;
    select new cut edges;
    Concorde(C, $F_E \setminus F_U$); // TSP Solver
  else
    break;
  end if
end while

4.3 Selection of Multiple Cut Edges

After the above iterative improvement procedure, a scan chain with ECO FFs will use only one cut edge. By using multiple cut edges, we can further reduce the total WL. Multiple cut edges can be viewed as multiple depots in a classical VRP. We solve this IMSCO problem as an instance of MDVRP with *dynamic depots*, that is, the location of the depots and the number of depots are not fixed. Algorithm 3 shows the selection of multiple cut edges using MDVRP. Based on multi-way clustering of ECO FFs and assignment of clusters to original scan chains, we use a *gain* calculation to guide the selection of multiple cut edges according to Equation (2). Given a cluster of ECO FFs, the gain from separating the cluster into two sub-clusters is found by disconnecting the two longest edges in the cluster's scan chain c_k, then reconnecting each

of the sub-clusters to the nearest original scan FF in c_k. (Edges incident to a nearest original scan FF are candidates to become cut edges.) If this results in a smaller total scan chain cost, we say that there is a *positive gain* from the operation. We calculate gain for all clusters in all scan chains, then implement the cluster separation that achieves the maximum gain. We continue to divide clusters into two, and select new cut edges for the two subclusters, until the maximum gain is no longer positive. We then perform ordering with the newly selected cut edges.

4.4 Ordering ECO FFs

We can formulate ordering of ECO FFs as an instance of classical TSP (cf. Section 2). We use the TSP solver, *Concorde*, to order ECO FFs in scan chains. *Concorde* implements the cutting-plane method, and uses a branch-and-cut search scheme [3]. On more complicated instances, if the cutting-plane method cannot find the optimal solution, *Concorde* switches to a branch-and-bound search.[2] It uses an independent LP solver software to implement the cutting-plane algorithm. In each phase of our algorithm, *Concorde* is executed to find the optimal or near-optimal wirelengths.

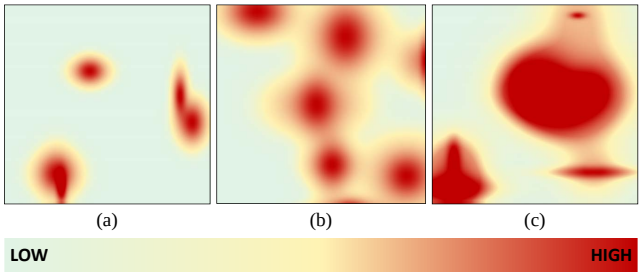


Figure 5: Examples of congestion maps from our congestion map generator. These three congestion maps are used to obtain the results given in Table 8.

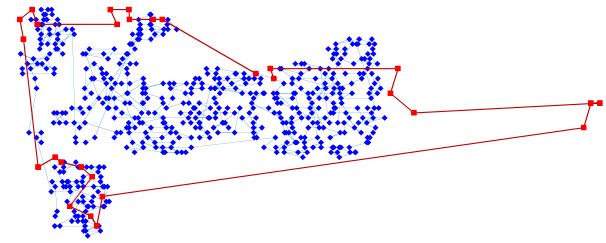
Congestion-aware ordering. To avoid disruption of existing routing and timing closure, we consider congestion information as well. We use congestion maps (using our congestion map generator as shown in Figures 5(a) – (c), or from commercial P&R tools) that consist of grid-cell arrays. Each grid cell contains the number of available routing tracks.

5. VALIDATION AND RESULTS

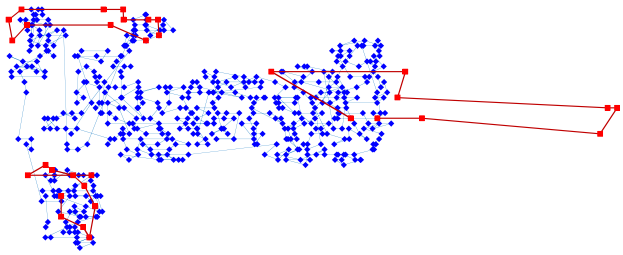
Our *ISC-solver* code implements our three-phase algorithm presented in C++ and is compiled with g++ 4.8.0. We use the TSP solver *Concorde* [2] to order scan FFs. For congestion-aware ordering, we use the 3-opt algorithm in *Concorde* with our congestion-aware distance function in Equation (3).

ISC-solver is validated on a 2.5GHz Intel Xeon E5-2640 Linux workstation with 128GB memory and 12 hyperthreaded CPU cores. *ISC-solver* is configurable with several user-specified options such as, maximum scan depth, maximum and minimum edge lengths between two flip-flops, and maximum number of cut edges for both chain and entire design. To verify *ISC-solver*, we use two kinds of testcases: (1) industrial, and (2) randomly generated using our scan instance generator. The scan instance generator creates testcases as well as congestion maps. The generator is also configurable, with user-specified options including layout size, number of scan chains, physical information of scan FFs in scan chains, number of ECO FFs, (x, y) locations of ECO FFs, compatible scan chain sets, distribution type of scan FFs, congestion map, size of grid cells, and congestion values in grid cells. We generate our results with $\alpha = \beta = \gamma = 0.33$ in Equation (1).

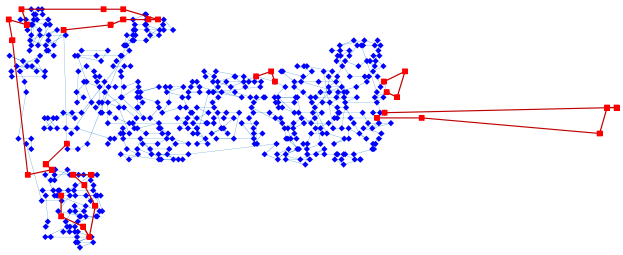
[2] Sometimes *Concorde* provides different ordering results. As an example, for an input with 634 cities and geometric distances, we divide the cities into 34 groups and run *Concorde* 50 times. We find the results are normally distributed with $\mu = 22769.64$ and $\sigma = 67.05$, and can vary up to 1.5% from the minimum result. To denoise, in all results reported below we repeat each run of *Concorde* 50 times and record the best result.



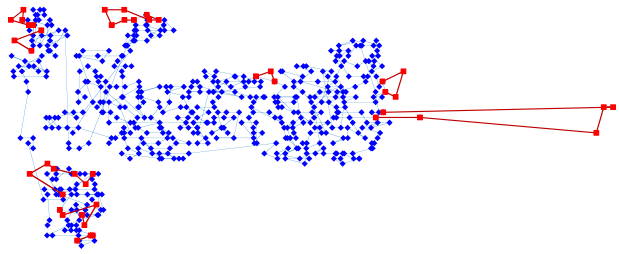
(a) Final chain WL = **4115.58μm** when $mce = 1$.



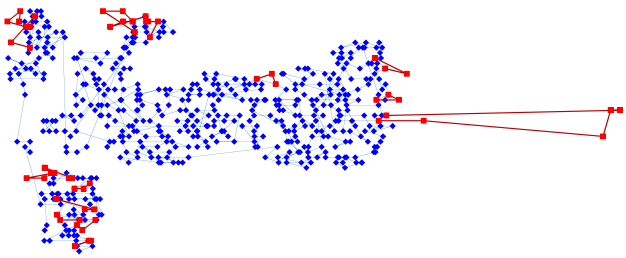
(b) Final chain WL = **3949.82μm** when $mce = 3$.



(c) Final chain WL = **3904.46μm** when $mce = 5$.



(d) Final chain WL = **3817.94μm** when $mce = 10$.



(e) Final chain WL = **3782.01μm** when $mce = \infty$.

Figure 6: Example non-congestion-aware solutions of an industrial scan chain solved by *ISC-solver*.

5.1 Comparison to Industrial Results

Table 2 compares scan depth SD achieved by *ISC-solver* to SD in the industrial results of manual ECO FF insertion. The industrial testcase contains 320 scan chains, 634 ECO FFs, and seven compatible scan chain groups. Table 2 compares change in SD across

the industrial testcase, where seven different ECOs were applied to a design. The maximum value of SD in the original industrial testcase (Ind in tables) is 574. For example, in the case of ECO1, manual ECO FF insertion changes SD to 592 from 562, that is, the manual solution requires up to 5.3% additional test time compared to the *ISC-solver* solution. Table 3 compares final WL between manual (industry) and *ISC-solver* solutions; *ISC-solver* achieves a 6593.8μm (45.71%) reduction in incremental wirelength. Further, *ISC-solver* (ISC in tables) does not increase the scan depth, meaning that no additional test time is required with our solution. Figures 6(a) – (e) show example solutions of an industrial scan chain solved by *ISC-solver* with different *mce* values. The red blocks and blue diamonds indicate ECO FFs and original scan FFs, respectively. Clearly, for larger ECOs, our algorithm performs better than manual solutions.

Table 2: Comparison of final scan depth (SD).

Group	Orig Max SD	#ECO FF	Ind SD	ISC SD	Ind ΔSD	ISC ΔSD
ECO1	562	508	**592**	562	**30 (5.3%)**	**0 (0.0%)**
ECO2	565	77	565	565	0 (0.0%)	**0 (0.0%)**
ECO3	373	3	373	373	0 (0.0%)	**0 (0.0%)**
ECO4	565	7	565	565	0 (0.0%)	**0 (0.0%)**
ECO5	471	6	476	471	**5 (1.1%)**	**0 (0.0%)**
ECO6	574	2	574	574	0 (0.0%)	**0 (0.0%)**
ECO7	573	30	576	573	**3 (0.5%)**	**0 (0.0%)**

Table 3: Comparison of final wirelength (WL) (μm).

Orig	Ind	ISC	Ind – Orig	ISC – Orig
1984806.7	1999232.3	1992638.5	14425.6	7831.8 (**–45.71%**)

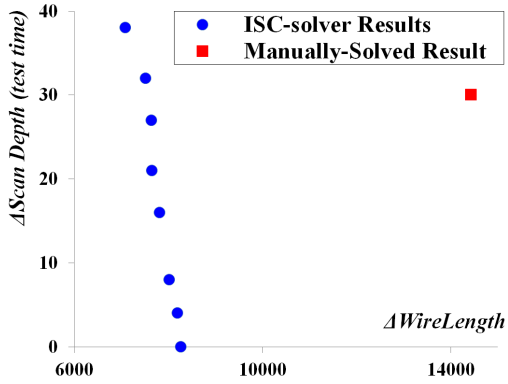


Figure 7: $\Delta ScanDepth$ vs. $\Delta WireLength$, ISC-solver vs. industry.

5.2 Scalability of Our Implementation

ISC-solver is configurable with several user-specified options. Table 5 presents results for the industrial testcase with various configurations of user-specifiable options. In the table, *psd* is the maximum percentage increase in scan depth allowed after ECO FFs are inserted into a compatible scan chain set; *lel* and *mel* are respectively the lower and upper bounds for edge lengths between two scan FFs; *mce* and *MCE* are respectively the maximum allowed number of cut edges for each scan chain and for the entire design. The original scan chain has SD = 574 and WL = 1984806.64μm. By increasing *psd*, we observe that the final scan depth increases. If *psd* = 0, the final scan depth cannot be increased beyond the original scan depth of 574. If *lel* is increased, the final wirelength and the number of unwired ECO FFs increase. When *mel* = 200, *ISC-solver* provides the largest benefit with respect to the final wirelength, the number of cut edges, and the number of unwired ECO FFs. By increasing *mce* and *MCE*, the final wirelength and the number of unwired ECO FFs are both reduced, since more cut edges are allowed.

ISC-solver has $O(mn^2)$ time complexity for each iteration of affinity vector calculation, where m is the number of original scan FFs and n is the number of ECO FFs. To demonstrate scalability of *ISC-solver*, we conduct experiments with two randomly generated testcases. The first testcase has 100 scan chains with scan depths between 400–600, a random distribution of ECO FFs, and a congestion map. The second testcase contains {20, 50, 100, 200, 500} original scan chains, 300 ECO FFs, and a congestion map. The top and bottom halves of Table 4 respectively show the scalability of *ISC-solver* across these two testcases. These runtime and memory data are representative of *ISC-solver* trends when #scan chains is fixed and #ECO FFs varies (top half of figure), and when #ECO FFs is fixed and #scan chains varies (bottom half of figure).

Table 4: Scalability of *ISC-solver*.

#ECO FFs	#Scan Chains	#Orig Scan FFs	CPU time (s)	Mem (MB)
100	100	49300	11.56	28.99
250	100	49300	52.12	29.14
500	100	49300	206.66	29.51
1000	100	49300	801.00	30.70
2500	100	49300	4766.29	35.07
300	20	9832	131.76	56.64
300	50	25693	354.77	62.88
300	100	50002	673.99	72.26
300	200	99203	1299.38	91.42
300	500	250059	3025.67	149.25

5.3 Impact of Congestion-Aware Cost Function

ISC-solver is able to cluster and order ECO FFs guided by a user-provided congestion map. We conduct experiments with three different congestion maps, {200, 500, 1000, 2000} ECO FFs, and 500 original scan chains with #FFs between 400–600. Table 8 shows results for the randomly generated testcases and congestion maps from Figures 5(a) – (c). We observe that post-ECO WL values with and without congestion are clearly different for the same testcase, and that congestion-aware clustering and ordering does not consume significantly more system resources. Figures 8(a) and 8(b) respectively show non-congestion-aware and congestion-aware solutions for a small example containing 200 ECO FFs. The figures show how *ISC-solver* avoids highly congested regions during initial clustering and ordering.

Table 6: Incremental WL (in μm) for solutions and costs with and without congestion awareness (representative result).

		Cost	
		w/o congestion	w/ congestion
Solution	w/o congestion	**72280.09**	117143.14
	w/ congestion	84470.61	**100516.30**

To further illustrate the "sanity" of congestion-aware solutions, Table 6 gives representative results of WL for four distinct treatments of a testcase: (i) when both solution and cost function do not consider congestion; (ii) when both solution and cost function consider congestion; (iii) when only solution considers congestion; and (iv) when only cost function considers congestion. As expected, WL values satisfy (i) $<$ (iii), and (ii) $<$ (iv). These representative results illustrate intuitively reasonable behavior of *ISC-solver*'s solutions and cost functions.

Table 7: Impact of granularity of grid cell (Gcell) size.

Gcell X×Y	SD	WL (μm)	#CE	CPU Time (s)	Mem (MB)
10×10	600	2515210.74	264	344.49	83.77
20×20	600	2524434.69	241	347.45	89.53
50×50	600	2540594.35	249	386.16	322.12
100×100	600	2548770.12	243	1383.44	3900.93
200×200	600	2582814.58	230	7237.27	61123.81

Table 5: Industrial testcase results with various *ISC-solver* configurations. (SD = scan depth; #CE = number of cut edges.)

QoR	Configuration Options					Post-ECO with *ISC-solver*				System Resources	
	psd	**lel**	**mel**	**mce**	**MCE**	**SD**	**WL (μm)**	**#CE**	**#Unwired FFs**	**CPU time (s)**	**Mem (MB)**
psd	**1**	0	500	∞	10000	**578**	1993022.96	84	0	242.49	69.33
	2	0	500	∞	10000	**582**	1992830.60	104	0	254.15	69.46
	5	0	500	∞	10000	**590**	1992321.28	106	0	272.28	69.46
	9	0	500	∞	10000	**612**	1991673.92	146	0	187.66	69.85
lel	0	**1**	500	∞	10000	574	**1994232.28**	69	**0**	246.80	69.21
	0	**2**	500	∞	10000	574	**1995079.84**	62	**0**	234.42	69.59
	0	**5**	500	∞	10000	574	**1997388.72**	94	**0**	215.81	69.20
	0	**10**	500	∞	10000	574	**2002900.52**	102	**1**	245.17	69.33
	0	**20**	500	∞	10000	574	**2012828.76**	86	**2**	292.19	69.20
	0	**50**	500	∞	10000	574	**2039497.64**	110	**99**	215.43	69.21
mel	0	0	**50**	∞	10000	574	**1990935.84**	103	**74**	139.23	69.60
	0	0	**100**	∞	10000	574	**1993875.00**	84	**7**	206.17	69.34
	0	0	**200**	∞	10000	574	**1992638.52**	73	**0**	181.88	69.21
	0	0	**500**	∞	10000	574	**1993001.68**	69	**0**	163.98	69.33
	0	0	**1000**	∞	10000	574	**1993545.16**	113	**0**	158.26	69.33
	0	0	**2000**	∞	10000	574	**1993765.56**	98	**0**	167.25	69.33
mce	0	0	5000	**1**	10000	574	**1997098.36**	**35**	0	231.21	69.20
	0	0	5000	**3**	10000	574	**1994494.64**	**56**	0	216.85	69.20
	0	0	5000	**5**	10000	574	**1994317.12**	**67**	0	214.82	69.21
	0	0	5000	**10**	10000	574	**1993972.72**	**83**	0	237.14	69.33
	0	0	5000	**20**	10000	574	**1993633.64**	**102**	0	176.41	69.33
MCE	0	0	5000	∞	**10**	574	1988055.76	**10**	**417**	84.14	69.33
	0	0	5000	∞	**20**	574	1994268.40	**20**	**104**	167.88	69.60
	0	0	5000	∞	**30**	574	1996570.28	**30**	**4**	201.18	69.86
	0	0	5000	∞	**50**	574	1994547.28	**48**	**0**	185.19	69.21
	0	0	5000	∞	**100**	574	1993683.76	**89**	**0**	183.03	69.33

Table 7 compares solution quality in a $25mm^2$ layout when the number of grid cells is varied. When *ISC-solver* imports the congestion map, *ISC-solver* internally coalesces multiple adjacent tool-reported small grid cells into larger grid cells (*Gcells* in Table 7). To reduce the runtime, *ISC-solver* stores congestion coefficients (i.e., x_{Con} and y_{Con}) between two scan FFs. The testcase consists of 200 original scan chains with the maximum scan chain depth of 600 and 500 ECO FFs. Solution quality measured in terms of WL, #CE and SD are similar. However with 100×100 or more grid cells, the runtime and memory usage increases by $\sim3\times$ and $\sim10\times$ respectively as compared to with 10×10 grid cells. This is because with 100×100 or more grid cells, several congestion coefficients must be computed at runtime and few coefficients can be reused between grid cells, whereas with 10×10 grid cells most congestion coefficients can be precalculated and reused.

6. CONCLUSIONS

To improve testability of ECO logic in the SOC implementation flow, incremental multiple-scan chain ordering for ECO FF insertion can efficiently support large front-end RTL modifications when the design flow is near tapeout stage. To avoid loss of test coverage, ECO FFs must be added into existing scan chains with maximum preservation of routing and timing closure. We present a new *Incremental Multiple-Scan Chain Ordering* formulation as the basis for future automation of the tedious and time-consuming process of manual scan stitching. Our proposed heuristic approach consists of clustering, incremental clustering and ordering steps to minimize the maximum chain length (test time), routing congestion, and disturbance to existing scan chains. Our implementation, *ISC-solver*, shows dominant performance compared to physical design engineers' manual solutions for an industrial design. *ISC-solver* successfully minimizes the excess scan chain length, the incremental wirelength, and the disruption of existing scan chains. With its heuristic approach based on greedy construction and iterative improvement, *ISC-solver* achieves 5.3% test time reduction (i.e., no additional test time from the ECO), and 45.71% reduction of incremental wirelength, versus the manual (production tapeout) solution for the industrial testcases. *ISC-solver* can also consider congestion information to efficiently avoid high-congestion areas in the chip layout. Various user-specified options allow tuning of the solution to various objectives. We conclude that *ISC-solver* provides a promising approach to dealing with the insertion of ECO FFs into an existing design which is near tapeout. Our ongoing work pursues various code optimizations (speedups) as well as better connections to the operations research literature, e.g., via the dynamic multi-depot VRP with variable number of movable depots.

Acknowledgments

We thank Mr. Ching-Yao Liu for his participation in our initial investigations. We also thank Ms. Nancy MacDonald for valuable feedback and discussions of problem formulations.

7. REFERENCES

[1] M. Abramovici, M. A. Breuer and A. D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, 1990.

[2] D. L. Applegate, R. E. Bixby, V. Chvátal and W. Cook, "Concorde TSP Solver", *http://www.tsp.gatech.edu*

[3] D. L. Applegate, R. E. Bixby, V. Chvátal, W. Cook, D. G. Espinoza, M. Goycoolea and K. Helsgaun, "Certification of an Optimal TSP Tour through 85,000 Cities", *OR Letters* 37(1) (2009), pp. 11-15.

[4] T. Bektas, "The Multiple Traveling Salesman Problem: An Overview of Formulations and Solution Procedures", *Omega* 34(3) (2006), pp. 209-219.

[5] G. B. Dantzig and J. H. Ramser, "The Truck Dispatching Problem", *Management Science* 6(1) (1959), pp. 80-91.

[6] J.-F. Cordeau, M. Gendreau and G. Laporte, "A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems", *Networks* 30(2) (1998), pp. 105-119.

[7] M. Elm, H.-J. Wunderlich, M. E. Imhof, C. G. Zoellin, J. Leenstra and N. Maeding, "Scan Chain Clustering for Test Power Reduction", *Proc. DAC*, 2008, pp. 828-833.

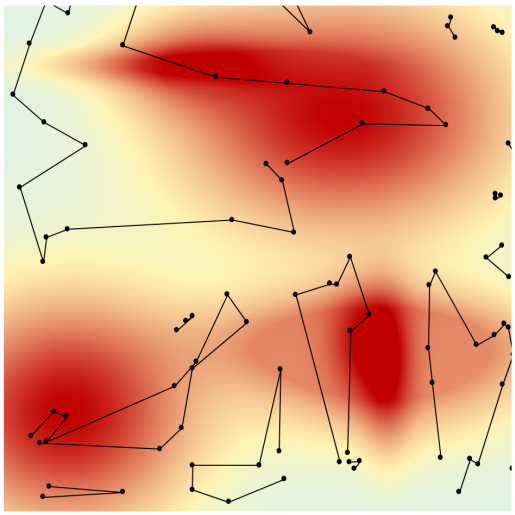
[8] M. Feuer and C. C. Koo, "Method for Rechaining Shift Register Latches Which Contain More Than One Physical Book", *IBM Technical Disclosure Bulletin* 25(9) (1983), pp. 4818-4820.

[9] C. M. Fiduccia and R. M. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions", *Proc. DAC*, 1982, pp. 175-181.

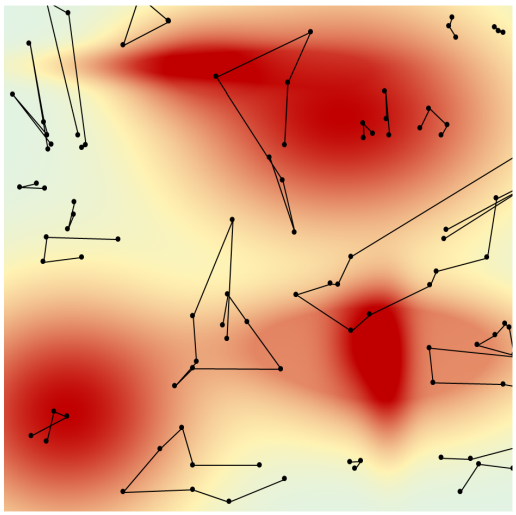
[10] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.

Table 8: Randomly generated testcase results with and without congestion information. (SC = #scan chains; SD = scan depth; #CE = number of cut edges.)

Congestion Aware?	Original Scan Information			#ECO FFs	Post-ECO Scan Information				
	SC	SD	WL (μm)		SD	WL (μm)	#CE	CPU Time (s)	Mem (MB)
No	500	600	5331786.45	200	600	5350960.45	19	18.61	93.30
				500	600	5354949.67	20	85.51	93.82
				1000	600	5379113.05	28	328.33	94.86
				2000	600	5400818.87	32	1300.72	97.84
Yes Figure 5(a)	500	600	5331786.45	200	600	5355131.92	24	47.10	133.78
				500	600	5362603.72	23	212.18	134.28
				1000	600	5387926.79	39	765.43	135.36
				2000	600	5434472.62	29	3280.43	140.04
Yes Figure 5(b)	500	600	5331786.45	200	600	5355222.36	20	52.96	133.91
				500	600	5362192.94	37	209.93	134.41
				1000	600	5396720.22	26	829.11	135.86
				2000	600	5429455.72	37	3318.68	139.37
Yes Figure 5(c)	500	600	5331786.45	200	600	5355800.32	23	47.96	133.78
				500	600	5366554.20	27	267.38	134.79
				1000	600	5398157.62	37	749.31	135.48
				2000	600	5443891.86	31	2983.94	139.19



(a) Non-congestion-aware solution.



(b) Congestion-aware solution.

Figure 8: Contrast between non-congestion-aware and congestion-aware *ISC-solver* solutions. A pair of path endpoints in each solution represents a cut edge in the original scan chain.

[11] M. Gendreau, J.-Y. Potvin, O. Bräysy, G. Hasle, and A. Løkketangen, *Metaheuristics for the Vehicle Routing Problem and its Extensions: A Categorized Bibliography*, Springer, 2008.

[12] P. Gupta, A. B. Kahng and S. Mantik, "Routing-Aware Scan Chain Ordering", *Proc. ASP-DAC*, 2003, pp. 857-862.

[13] P. Gupta, A. B. Kahng and S. Mantik, "A Proposal for Routing-Based Timing-Driven Scan Chain Ordering", *Proc. ISQED*, 2003, pp. 339-343.

[14] W. Ho, G. T. S. Ho, P. Ji and H. C. W. Lau, "A Hybrid Genetic Algorithm for the Multi-Depot Vehicle Routing Problem", *Engineering Applications of AI* 21(4) (2008), pp. 548-557.

[15] D. S. Johnson, "Local Optimization and the Traveling Salesman Problem", *Proc. ICALP*, 1990, pp. 446-461.

[16] G. Laporte, Y. Nobert and D. Arpin, "Optimal Solutions to Capacitated Multidepot Vehicle Routing Problems", *Congressus Numerantium* 44 (1984), pp. 283-292.

[17] G. Laporte, "The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms", *European Journal of Operations Research* 59(3) (1992), pp. 345-358.

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

[19] M. Mirabi, S. M. T. F. Ghomi and F. Jolai, "Efficient Stochastic Hybrid Heuristics for the Multi-Depot Vehicle Routing Problem", *Robotics and Computer-Integrated Manufacturing* 26(6) (2010), pp. 564-569.

[20] A.-S. Pepin, G. Desaulniers, A. Hertz and D. Huisman, "A Comparison of Five Heuristics for the Multiple Depot Vehicle Scheduling Problem", *Scheduling* 12(1) (2009), pp. 17-30.

[21] K. Rahimi and M. Soma, "Layout Driven Synthesis of Multiple Scan Chains", *IEEE Trans. on CAD* 22(3) (2003), pp. 317-326.

[22] J.-C. Rau, C.-H. Lin and J.-Y. Chang, "An Efficient Low-Overhead Policy for Constructing Multiple Scan-Chains", *Proc. Asian Test Symposium*, 2004, pp. 82-87.

[23] G. Reinelt, "TSPLIB95", *Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR)*, Heidelberg, 1995.

[24] J. Renaud, G. Laporte and F. F. Boctor, "A Tabu Search Heuristic for the Multi-Depot Vehicle Routing Problem", *Computers and OR* 23(3) (1996), pp. 229-235.

[25] G. Seok, I.-S. Lee, T. Ambler and B. F. Womack, "An Efficient Scan Chain Partitioning Scheme with Reduction of Test Data under Routing Constraint", *Proc. Symp. on Defect and Fault-Tolerance*, 2006, pp. 145-156.

[26] D. Xiang, M.-J. Chen, J.-G. Sun and H. Fujiwara, "Improving Test Effectiveness of Scan-Based BIST by Scan Chain Partitioning", *IEEE Trans. on CAD* 24(6) (2005), pp. 916-927.

[27] ScanOpt. *http://vlsicad.ucsd.edu/GSRC/Bookshelf/Slots/ScanOpt/* .