Mask-Cost-Aware ECO Routing*

Hsi-An Chien†, Zhen-Yu Peng†, Yun-Ru Wu‡, Ting-Hsiung Wang‡,
Hsin-Chang Lin‡, Chi-Feng Wu‡, and Ting-Chi Wang†
†Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan 30013
‡Realtek Semiconductor Corp., Hsinchu, Taiwan 30013

***Abstract*—In this paper, we study a mask-cost-aware routing problem for engineering change order (ECO). By taking into account old routes for possible reuse, we present an approach for the problem. Encouraging experimental results are reported to demonstrate the effectiveness of our approach.**

I. Introduction

Engineering change order (ECO) plays an important role during circuit design for fixing functional/timing errors or even for minor changing the specification. ITRS also pointed out that photomask cost is increasing dramatically as technology advances [2], such that post-silicon ECO implementation will affect the total manufacturing cost significantly. In the past few years, ECO-related problems were studied extensively, e.g., in [4], [3], [5]. However, there is only one published work which takes mask re-spin cost of routing layers into account, but its ECO routing method is not explained with enough details [4].

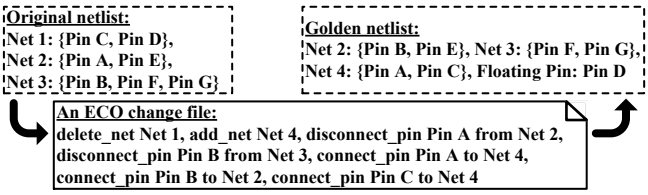


Fig. 1. An ECO implementation.

Generally, an ECO routing problem asks to find a routing solution for ECO nets that are produced by an ECO implementation. An ECO implementation considered in this paper has four types of modifications for a circuit: *net addition*, *net deletion*, *pin connection*, and *pin disconnection*. As an example given in Fig 1, a net addition (deletion) is to add (delete) a net logically, and a pin connection (disconnection) is to connect (disconnect) a pin to (from) a net logically. A net deletion can be also considered to disconnect all pins from a net. It is interesting to note that after some nets in the original netlist are modified, their original routing results will become *old routes* and be no longer valid. See Fig. 2 for an example, where by doing three pin disconnections and two pin connections on Net 1 and Net 2, the original netlist is modified and then an ECO routing problem for Net 1 and Net 2 in the golden

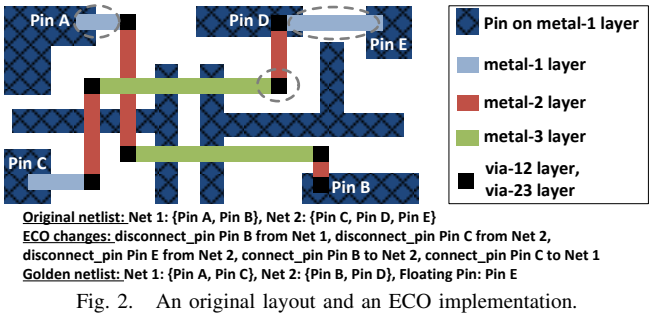


Fig. 2. An original layout and an ECO implementation.

*This work was supported in part by Realtek Semiconductor Corp. and National Science Council of Taiwan under Grant No. NSC-102-2220-E-007-012.

978-3-9815370-2-4/DATE14/©2014 EDAA

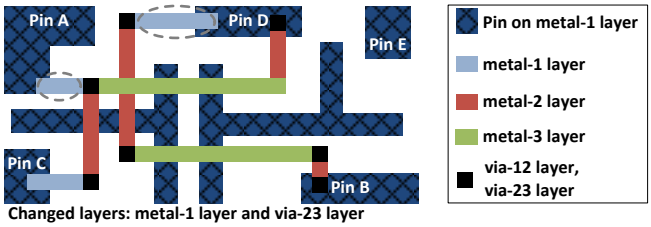


Fig. 3. A mask-cost-aware ECO routing result of Fig. 2.

netlist is created, while the old routes of the two nets become invalid. Intuitively, we can first remove old routes and then route ECO nets, but in the worst scenario, the resultant ECO routing solution may cause all metal and via layers to change, making the mask re-spin cost unacceptably high.

A post-silicon ECO routing problem is not trivial because it has to consider mask re-spin cost when using limited routing resources to achieve ECO implementation. If we smartly reuse some *routing objects* of the old routes during ECO routing, we can not only obtain additional routing resources but also reduce mask cost. Note that a routing object could be a metal segment or a via. Fig. 3 shows a mask-cost-aware ECO routing result that reuses some parts of old routes to realize the ECO implementation of Fig. 2. It removes the metal-1 routing object connected to Pin A, the metal-1 routing object connected to Pin D and Pin E, and a via-23 routing object on the old route of Net 2 (see the regions highlighted by dashed circles in Fig.2), reuses the remaining routing objects on the old routes, and adds the metal-1 routing object connected to Pin D and the metal-1 routing object connected to Pin A (see the regions highlighted by dashed circles in Fig. 3). This ECO routing result only changes metal-1 layer and via-23 layer while keeping via-12 layer, metal-2 layer, and metal-3 layer intact.

Motivated by the above example, we study in this paper a mask-cost-aware ECO routing problem. We present an approach to reuse old routes to accomplish post-silicon ECO routing and minimize total mask re-spin cost. Encouraging experimental results are reported to demonstrate the effectiveness of our approach.

II. Problem Formulation

For each net that is in the original netlist but gets modified by an ECO implementation, its old route is called a *reusable route*. A reusable route can be partially reused for routing ECO nets in the golden netlist.

For a post-silicon design, let $ML = \{ml_1, ml_2, ..., ml_n\}$ denote the set of n metal layers, $CL = \{cl_1, cl_2, ..., cl_{n-1}\}$ the set of $n-1$ via layers, and $C = \{c_{ml_1}, c_{cl_1}, c_{ml_2}, ..., c_{cl_{n-1}}, c_{ml_n}\}$ the set of $2n-1$ mask re-spin costs for these metal and via layers. For an ECO implementation, it contains a set of net deletions, a set of pin disconnections, a set of net additions, and/or a set of pin connections to modify the original netlist. These modifications produce a set EN of ECO nets to be routed, a set FP of pins to become floating, and a set RW of reusable routes.

In order not to disturb the nets which are not involved in an ECO implementation, their routes remain intact and

become obstacles when routing ECO nets. The available routing resources (including those occupied by reusable routes) are specified by a routing grid graph, where the vertices and edges that are occupied by obstacles are all removed. The mask-cost-aware ECO routing problem is formally defined as follows.

Problem 1 (Mask-Cost-Aware ECO Routing): Given EN, FP, RW, and the routing grid graph of a design, the objective of the mask-cost-aware ECO routing problem is to make all pins in FP become floating, route as many nets in EN as possible, and minimize the total mask re-spin cost induced by the set of changed metal and via layers.

III. Our Approach

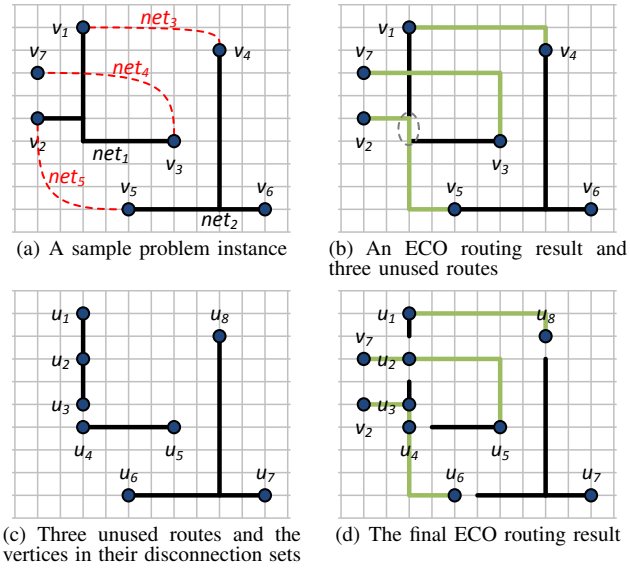
A. Overview

Let us first use Fig. 4 to give an overview of our approach. Fig. 4(a) shows an instance of our ECO routing problem, where each solid blue vertex denotes a pin, each tree composed of solid black line segments and pins denotes a reusable route, and each dashed red curve denotes an ECO net to be routed. It also shows that an ECO implementation changes the original netlist including two nets, $net_1 = \{v_1, v_2, v_3\}$ and $net_2 = \{v_4, v_5, v_6\}$, and one floating pin v_7 to the golden netlist including three ECO nets, $net_3 = \{v_1, v_4\}$, $net_4 = \{v_3, v_7\}$ and $net_5 = \{v_2, v_5\}$, and one floating pin v_6.

Our approach first enters *the ECO routing phase* which completes the routing for the three ECO nets by reusing routing objects on reusable routes, and by allowing *shorts* to occur between an ECO route and an *unused route* (see Fig. 4(b) where each tree in the green color shows the routing result for an ECO net, each tree in the black color shows an unused route, and the green line segment marked by the dashed circle is originally on the old route of net_1 but is now reused by the route of net_5). Note that: (1) An unused route is a tree that is a part of a reusable route but is not reused by any ECO net. (2) A short occurs when an ECO route intersects with an unused route at a point.

Our approach then enters *the connectivity-breaking phase* which disconnects all pins in FP from their original nets and removes all shorts. Intuitively, we can remove all unused routes to complete the goal of this phase. However, when the ECO routing result only causes a small set of layers to change, removing all unused routes could create changes for all mask layers (if unused routes scatter on all layers) and induce expensive mask cost. Instead, our approach first generates a set, called *disconnection set*, from each unused route; each element in the set is called a *disconnection vertex* and corresponds to a vertex in the routing graph that either is an intersection point of the unused route and an ECO route, or is occupied by a floating pin in FP. For example, there are three unused routes in Fig. 4(b), so our approach will generate three disconnection sets, $D_1 = \{u_1, u_2, u_3\}$, $D_2 = \{u_4, u_5\}$, and $D_3 = \{u_6, u_7, u_8\}$, as shown in Fig. 4(c) (note that v_1, v_3, v_4, v_5, v_6 are renamed as u_1, u_5, u_8, u_6, u_7, respectively). For each disconnection set, our approach continues to delete some routing objects on the corresponding unused route such that each vertex is physically disconnected from each other in the set. After doing this, all shorts are eliminated and all pins in FP become floating (see Fig. 4(d) for the final ECO routing result, where some routing objects on unused routes are deleted to make all vertices in each D_i become disconnected, $i = 1, 2, 3$).

Our approach will consider the total mask re-spin cost in both the ECO routing phase and the connectivity-breaking phase. The details of the two phases are presented in the following two subsections



(a) A sample problem instance

(b) An ECO routing result and three unused routes

(c) Three unused routes and the vertices in their disconnection sets

(d) The final ECO routing result

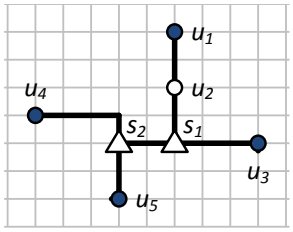
Fig. 4. Illustration of our approach.

B. ECO Routing Phase

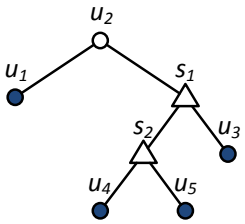
The ECO routing phase first decomposes each ECO net into a set of 2-pin sub-nets according to the minimum spanning tree of the ECO net, and determines a set L of metal layers by collecting the layers on which the pins of ECO nets are located. It then performs ECO routing using L and the via layers in between as *the layer range*. It is forbidden to create new routing objects outside the layer range, but our approach can still utilize the routing objects of reusable routes on layers outside the layer range whenever necessary. If there are some ECO nets that cannot be successfully routed (called *failed nets*), the ECO routing phase will iteratively add an adjacent layer of a lower mask cost to L to enlarge the layer range and reroute all ECO nets, until all ECO nets are successfully routed or no more layer can be added. In the rest of this subsection, the details of our ECO routing algorithm under a fixed layer range are described.

For a given routing layer range, our ECO routing algorithm is comprised of two stages. In the first stage, it routes each ECO net one by one. The ECO nets are routed in a non-decreasing order of their minimum bounding-box sizes while each ECO net is routed in a sub-net by sub-net manner. The sub-nets of each ECO net are routed in a random order. The routing graph consists of all available routing resources in the layer range as well as those occupied by reusable routes. To encourage using reusable routes for ECO routing, their routing edges are assigned a cost of 0, while the other edges are set to a cost of 1 at the beginning. Furthermore, to encourage the sub-nets belonging to the same ECO net to share routing resources, two or more of them passing through an edge e will not cause any overflow to e, and the cost of e is set to 0 for these sub-nets except for the first one. Since the shapes of pins are irregular in advanced technology nodes, it is not uncommon for a pin to occupy some vertices in the routing grid graph. Hence, we generalize Dijkstra's algorithm to get a multi-source multi-target shortest path algorithm for routing each 2-pin sub-net. When the shortest path algorithm reaches any one of the target vertices, it will stop and return the found shortest path.

In the first stage, overflow on any edge is prohibited, and our algorithm iteratively expands the routing region from the minimum bounding box of each failed ECO net until all ECO nets are successfully routed, or it reaches the iteration limit but still fails to route all ECO nets. When the latter case



(a) disconnection vertices: $u_1 \sim u_5$; Steiner points: s_1 and s_2



(b) Topology tree rooted at u_2

$$ses_1^{u_2} = \{e_{u_2u_1}, e_{u_2s_1}\}, \quad ses_2^{u_2} = \{e_{u_2u_1}, e_{s_1s_2}, e_{s_1u_3}\}, \quad ses_3^{u_2} = \{e_{u_2u_1}, e_{s_1u_3}, e_{s_2u_4}, e_{s_2u_5}\}$$

(c) Separating sets for u_2

$$ALS(e_{u_2u_1}) = \{ml_1, cl_1\}, ALS(e_{u_2s_1}) = \{ml_1\}, ALS(e_{s_1s_2}) = \{cl_1, ml_2\}, ALS(e_{s_1u_3}) = \{cl_2\}, ALS(e_{s_2u_4}) = \{cl_1\}, ALS(e_{s_2u_5}) = \{ml_2\}$$

(d) Layer set of each tree edge

$$LC(ses_1^{u_2}) = \{\{ml_1\}, \{ml_1, cl_1\}\}, \quad LC(ses_3^{u_2}) = \{\{ml_1, cl_1, ml_2, cl_2\}, \{cl_1, ml_2, cl_2\}\},$$
$$LC(ses_2^{u_2}) = \{\{ml_1, cl_1, cl_2\}, \{ml_1, ml_2, cl_2\}, \{cl_1, ml_2, cl_2\}, \{cl_1, cl_2\}\},$$
$$LC(u_2) = LC(ses_1^{u_2}) \cup LC(ses_2^{u_2}) \cup LC(ses_3^{u_2})$$

(e) Collection of layer combinations of each separating set

Fig. 5. Illustration of topology tree, separating sets, and layer combinations.

happens, our algorithm enters the second stage to route each failed ECO net by allowing overflow to occur during the course of routing a sub-net. However, if the routing result of a sub-net causes an edge e to overflow, the cost of e will be incremented immediately and the previously routed sub-net that passes through e will be ripped up. Each sub-net that is ripped up or failed will be re-routed in the next iteration (note that in the second stage, a sub-net is said to be a failed sub-net if there exists no path in its current routing region to connect its two pins, even if overflow is allowed). If no improvement is obtained (i.e., the total number of failed and ripped-up sub-nets does not decrease) at the current iteration, our algorithm further expands the routing regions for each failed and ripped-up sub-nets before going to the next iteration; otherwise, it only expands the routing region for each failed sub-net. This stage will terminate when there is no failed or ripped-up sub-net, or it reaches a given iteration limit.

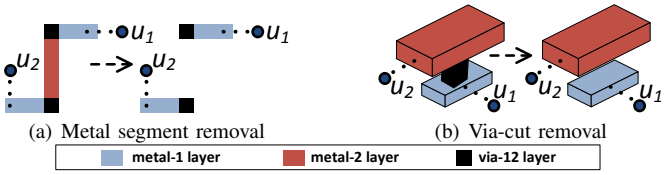
C. Connectivity-Breaking Phase

The connectivity-breaking phase solves the following problem to remove all shorts and make all pins in FP become floating.

Problem 2 (Mask-Cost-Aware Connectivity-Breaking): Given a set of unused routes each of which is associated with a disconnection set, the objective of the mask-cost-aware connectivity-breaking problem is to remove routing objects from each unused route for disconnecting all vertices in the associated disconnection set such that the total mask re-spin cost induced by the changed layers is minimized.

For each unused route, our connectivity-breaking algorithm first builds a set of *topology trees* each of which is rooted at a different disconnection vertex. Each tree node in a topology tree corresponds to a disconnection vertex or a Steiner point on an unused route, and each tree edge corresponds to a routing path on the unused route that connects the two nodes of the edge. For example, Fig. 5(a) gives an unused route that has five disconnection vertices u_1, u_2, u_3, u_4, u_5, and two Steiner points s_1, s_2; therefore for each u_i, a topology tree rooted at u_i can be built. One of the five topology trees is shown in Fig. 5(b), which is the corresponding topology tree rooted at u_2.

Then for each rooted topology tree, our algorithm traverses it to find a collection of *separating sets*. A separating set is a minimal subset of edges of a rooted topology tree such that after deleting all the edges of the set from the tree, the root is disconnected from the other disconnection vertices. For example, for the rooted topology tree in Fig. 5(b), three separating sets $ses_1^{u_2}$, $ses_2^{u_2}$, and $ses_3^{u_2}$ as shown in Fig. 5(c) are generated, where e_{u_i,u_j} denotes the edge connecting node u_i and node u_j.



(a) Metal segment removal (b) Via-cut removal

Fig. 6. Routing object removal.

For each separating set, our algorithm continues to find a collection of *layer combinations* from its edges. Recall that each edge e represents a path of an unused route, and the path consists of a set of routing objects. The *layer set* of e is the set of layers on which the routing objects associated with e are located. For each separating set, our algorithm enumerates all possible layer combinations from the layer set of each edge such that there is exactly one layer from each layer set in each layer combination. For example, we assume the layer set $ALS(e)$ of each edge e of the topology tree of Fig. 5(b) is shown in Fig. 5(d), where ml_i and cl_j denote metal layer i and via layer j, respectively. Then the collection of layer combinations, $LC(ses_i^{u_2})$, for each separating edge set $ses_i^{u_2}$ of Fig. 5(c) is shown in Fig. 5(e), where each element in $LC(ses_i^{u_2})$ is a set and represents a layer combination of u_2. Let $LC(u_2)$ denote the set of all layer combinations for u_2, which is the union of all $LC(ses_i^{u_2})$'s, as shown in Fig. 5(e). Each layer combination in $LC(u_2)$ is said to *cover* u_2 and is a layer set candidate to carry out the disconnection of u_2 from the other disconnection vertices. Note that different disconnection vertices may have one or more same layer combinations covering them; in other words, a layer combination may cover one or more disconnection vertices.

To disconnect a disconnection vertex u_i from the others, our algorithm will select one layer combination lc_j from $LC(u_i)$ and perform *metal segment removal* and/or *via-cut removal* in the layers of lc_j. Removing a metal segment only changes a single mask. However, removing a via could change the number of masks up to three due to via enclosure rules. Thus, our algorithm does not remove a whole via, but replaces the via with its two enclosing metals instead so that only the via layer gets changed. As can be seen from Fig. 6(a) and Fig. 6(b), we respectively remove a metal-2 segment and a via-cut so as to break the connectivity between u_1 and u_2.

Once all layer combinations for each disconnection vertex are generated, the mask-cost-aware connectivity-breaking problem can be reduced to the problem of selecting a collection of layer combinations such that every disconnection vertex is covered by at least one layer combination in the collection. The union of these selected layers combinations is a set of layers whose total mask cost should be as small as possible. Our approach transforms this selection problem into an integer linear program (ILP) which is then solved by an open-source ILP solver [1]. Please note that if a layer gets changed by the ECO routing phase, its mask cost is set to 0 here because it is free to use now. The ILP formulation is given as follows:

$$\text{Minimize} \quad \sum_{c_i \in C} c_i \times b_i$$
$$\text{subject to} \quad \sum_{k \in f(lc_j)} x_j \geq 1, \ k = 1, \ldots, m$$
$$b_i \geq x_j, \ \forall c_i \in C, \forall lc_j \ni i.$$

TABLE II
OUR ECO ROUTING RESULTS

Case	#DRC	#DRC Opt.	Initial Changed Layers	Final Changed Layers	#L	T(s)
ECO1	7	0	ml_1, cl_1, ml_2	ml_1, cl_1, ml_2	3	3
ECO2	26	0	ml_1, cl_1, ml_2	ml_1, cl_1, ml_2, cl_2	4	4
ECO3	1081	104	ml_1, cl_1, ml_2, cl_2, ml_3, cl_3, ml_4, cl_4, ml_5	ml_1, cl_1, ml_2, cl_2, ml_3, cl_3, ml_4, cl_4, ml_5	9	6180

TABLE III
ECO ROUTING RESULTS OF THE COMMERCIAL TOOL

Case	With Layer Range			Without Layer Range				
	#DRC	#DRC Opt.	T(s)	#DRC	#DRC Opt.	T(s)	Changed Layers	#L
ECO1	15	10	8	0	-	4	ml_1, cl_1, ml_2, cl_2, ml_3, cl_3, ml_4	7
ECO2	15	12	15	0	-	4	ml_1, cl_1, ml_2, cl_2, ml_3, cl_3, ml_4	7
ECO3	774	763	3540	580	588	410	ml_1, cl_1, ml_2, cl_2, ml_3, cl_3, ml_4, cl_4, ml_5, cl_5, ml_6	11

TABLE I
ECO CASES

Case	ECO1	ECO2	ECO3
#ECO Nets in EN	2	9	68
#Floating Pins in FP	0	9	0
#Resuable Routes in RW	2	13	27
Gate Count	19,872	19,872	1,034,892
#Nets	13,355	13,355	984,972
Utilization (%)	77.42	77.42	69.2
Core Area (μm*μm)	233*273	233*273	2,545*1,794

where c_i is the mask re-spin cost for each i, C is the set of costs for all layers, b_i is a binary variable ($b_i = 1$ if layer i is chosen for participation in the connectivity breaking), x_j is a binary variable to denote whether the layer combination lc_j is selected or not, k denotes a disconnection vertex numbered from 1 to m, and $f(lc_j)$ is the set of disconnection vertices covered by lc_j. The objective is to minimize the total mask re-spin cost while every disconnection vertex k is covered by a layer combination at least.

Note that we have also developed several techniques to reduce the size and the solution space of the ILP formulation without sacrificing the optimality. However, due to the page limit, their details are omitted.

IV. Experimental Results

We have implemented our ECO routing approach in C++ language. Since our approach only considers simple design rules such as wire width and wire spacing rules, we develop a methodology that first runs our approach and then runs a commercial routing tool to perform incremental routing optimization for further fixing other types of design rule violations for our ECO routing result. Note that incremental routing optimization is executed under the layer range produced by our approach. We also compare our methodology with this commercial tool. The commercial tool is run by performing its ECO routing function, followed by incremental routing optimization when the produced ECO routing result is not DRC-clean, on two modes. The first mode does not put any layer range while the second mode uses the same layer range reported by our approach. Note that unlike our approach, the commercial tool does not offer the option to automatically minimize the total cost of changed masks. That is the reason why we run the commercial tool on two modes for being able to compare it with our methodology.

Three industrial ECO routing cases are used, and their detailed information is shown in Table I. Every case has 6 metal layers in 55nm node. We set the cost of each mask the same in the experiments, so that the total mask re-spin cost is measured by the number of changed layers. All experiments were conducted on a Linux workstation with a 2.9 GHz CPU and 48G memory. The routing results of our methodology and the commercial tool are shown in Table II and Table III, respectively. In the two tables, we report the number of DRC violations (in the "#DRC" column), the number of DRC violations after performing incremental routing optimization by the commercial tool (in the "#DRC Opt." column), the set of layers that are changed by our ECO routing phase (in the "Initial Changed Layers" column), the set of changed layers after our connectivity-breaking phase (in the "Final Changed Layers" column), the set of changed layers after performing ECO routing and possibly incremental routing optimization by the commercial tool (in the "Changed Layers" column), the amount of layers for the set in the "Final Changed Layers" column or for the set in the "Changed Layers" column (in the "#L" column), and the total runtime measured in second for our methodology or the commercial tool (in the "T(s)" column).

As can be seen in Table II, all DRC violations can be cleaned eventually after applying the incremental routing optimization to our routing solutions for ECO1 and ECO2 cases, which means that our approach can produce a better initial routing for subsequent DRC violation cleaning. As for ECO3, the DRC violations are too many to be completely resolved by the commercial tool. For ECO2, our connectivity-breaking phase needs one more via layer cl_2 than the ECO routing phase to remove all shorts. For the other cases, our connectivity-breaking phase can remove all shorts within the layers that are changed by our ECO routing phase.

We next analyze the results of the commercial tool in Table III. Given the final changed layers produced by our approach as the layer range, the commercial tool cannot get a DRC-clean solution for every case, but our approach works well for two cases after incremental routing optimization. Furthermore, when we do not set any layer range, the commercial tool produces DRC-clean solutions, respectively for ECO1 and ECO2, and therefore we do not further do incremental routing optimization for the two cases. However, compared with our methodology, the commercial tool needs four and three more layers to get DRC-clean for ECO1 and ECO2, respectively. Apparently, our methodology saves about half the cost for both cases as compared to the commercial tool. For ECO3 without the layer range, the commercial tool cannot find a DRC-clean solution even when it uses up all the routing layers.

V. Conclusion

In this paper, we have presented an ECO routing approach that considers the reuse of old routes to save the mask re-spin cost. We are currently taking more complex design rules into account to further improve our approach.

References

[1] lp_solve: http://lpsolve.sourceforge.net.
[2] International Technology Roadmap for Semiconductors, 2009.
[3] H.-Y. Chang, I. H.-R. Jiang, and Y.-W. Chang. Simultaneous functional and timing eco. In *Proceedings of the Conference on Design Automation*, pages 140–145, 2011.
[4] S.-Y. Fang, T.-F. Chien, and Y.-W. Chang. Redundant-wires-aware eco timing and mask cost optimization. In *Proceedings of the International Conference on Computer-Aided Design*, pages 381–386, 2010.
[5] S.-L. Huang, C.-A. Wu, K.-F. Tang, C.-H. Hsu, and C.-Y. R. Huang. A robust eco engine by resource-constraint-aware technology mapping and incremental routing optimization. In *Proceedings of the International Conference on Computer-Aided Design*, pages 382–387, 2011.