MCFRoute 2.0 : A Redundant Via Insertion Enhanced Concurrent Detailed Router

Xiaotao Jia[†], Yici Cai[†], Qiang Zhou[†], Bei Yu[‡]
[†]Tsinghua National Laboratory for Information Science & Technology
[†]Department of Computer Science & Technology, Tsinghua University, Beijing, China
[‡]CSE Department, The Chinese University of Hong Kong, NT, Hong Kong
jxt11@mails.tsinghua.edu.cn,{caiyc,zhouqiang}@tsinghua.edu.cn,byu@cse.cuhk.edu.hk

ABSTRACT

In modern VLSI design, manufacturing yield and chip performance are seriously affected by via failure. Redundant via insertion is an effective technique recommended by foundries to deal with the via failure. However, due to the extreme scaling of feature size, it is more and more difficult to resolve redundant via insertion (RVI) with limited routing resource while obeying complicated design rules. In this paper, we propose an RVI enhanced concurrent detailed router, MCFRoute 2.0, which effectively avoids design rule violations through a compact integer linear programming (ILP) model. The proposed router can not only route all nets simultaneously but also search for redundant via positions for all via simultaneously during routing stage. In addition, it proposes an RVI aware pin access allocation to further improve the routing performance. Experimental results show that our detailed router outperforms an industry EDA tool that it improves the redundant via insertion rate by 21%, while reducing design rule checking violation count, total wire length and via count by 47%, 4% and 14%, respectively.

Keywords

Detailed Routing, ILP, Redundant Via Insertion, Design Rule

1. INTRODUCTION

In modern very large scale integrated (VLSI) circuit design, due to the extreme scaling of transistor feature size and the complicated design rules, ensuring manufacturing yield and chip reliability is facing more and more issues. Among all the issues, the via failure has a critical impact on chip performance, functionality, and manufacturing yield [1]. In circuits, a via is used to connect two components on two adjacent metal layers. A via may fail due to various reasons such as random defects, electromigration, cut misalignment, or thermal stress included voiding. A partial via failure may cause timing problem due to the increasing resistance of the via; while a complete via failure may lead to net open (unconnected) that will cause severe function error.

Redundant via (also known as double via) insertion is one of the most effective techniques recommended by foundries to provide via reliability. Through the redundant via insertion (RVI), a chip can not only reduce the via failure probability but also improve timing, especially for those critical nets [1]. In this paper, the initial single via is called *main via*.

GLSVLSI '16, May 18-20, 2016, Boston, MA, USA
© 2016 ACM. ISBN 978-1-4503-4274-2/16/05...$15.00
DOI: http://dx.doi.org/10.1145/2902961.2902966

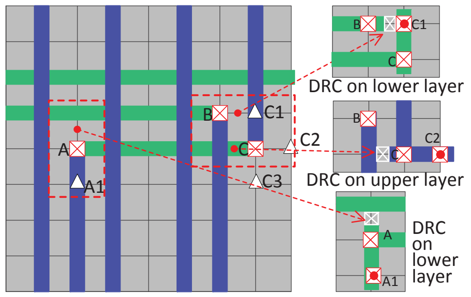

Figure 1: DRC violation by via enclosure.

The RVI problem has been well-studied in post-routing stage with the objective function to insert as many redundant vias as possible [2–7]. To further improve the via insertion rate, the RVI is also considered in early physical design stages [8–11]. Xu et al. [8] propose the first study on the RVI aware maze routing problem, which is transformed to a multi-constrained shortest path problem and solved by Lagrangian relaxation technique. Both [9] and [10] develop unified RVI aware routing frameworks consisting of global routing, detailed routing, and post-layout optimization. Lin et al. [11] propose an RVI aware track assignment and a dead-via aware detailed router.

However, very few of previous RVI aware routers consider practical via rule violations, thus they are divorced from reality. A *via enclosure* measures the distance from the cut array edge to the metal edge that encloses the cut array [12]. A conventional routing example with three vias is given in Fig. 1, where the white triangles represent the redundant via candidates. Different from [8], via *A* has no redundant via candidate if advanced design rules of via enclosure are considered. Meanwhile, via *C* only has one candidate. Take via *C* for example. If *C1* is redundant via of *C*, there is a spacing violation between it and via *B* on bottom layer; if *C2* is used, spacing violation is introduced on top layer. Therefore, only *C3* is a legal candidate. The white rectangles with cross line in the right figures represent design rule checking (DRC) violation markers. Although in this paper we assume each via has at most one redundant via candidate, the proposed techniques are generic enough that they can be naturally extended to the cases with multiple redundant vias.

Moreover, many existing detailed routing works are designed on a sequential manner, thus they suffer from the net-order problem that may affect the performance of final via insertion rate. By contrast, some routing works are implemented through a concurrent manner (e.g. [13,14]), which

Table 1: Notions of MCF Model

notion	meaning
$E/V/N$	edge/vertex/net set (indexed by $i/j/k$)
E_{v_j}	adjacent edges set of vertex v_j
$E_{v_j,out}$	edges set whose start point is vertex v_j
$E_{v_j,in}$	edges set whose end point is vertex v_j
$E^{\eta}_{v_j}$	via edges set of vertex v_j
$dg(v_j, d)$	the vertex degree of redundant via on direction d
$u(e_i)$	binary variable, the capacity of edge e_i
$c(e_i)$	positive variable, the cost of edge e_i
$d(n_k, v_j)$	the flow command of vertex v_j with value of -1, 0, or 1 that commodity n_k demands
$f(n_k, e_i)$	binary variable, flow of commodity n_k by edge e_i

overcomes the net-order problem and can get a global optimal solution in theory. In detailed routing area, Jia et al. [15] propose a concurrent methodology based on multi-commodity flow model and achieve good results. However, how to seamlessly integrate practical via design rules into the concurrent framework is still unclear.

In this paper, we present an RVI enhanced detailed router, which is following the concurrent method. To the best of our knowledge, this is the first work where complex via rules are considered in concurrent detailed router. The main contributions of this paper include:

- The proposed router is able to search for redundant via position for every via, while all the nets are routed simultaneously.
- We propose a comprehensive analysis on the effect of via enclosure, which is rarely discussed by previous academic works. It makes the proposed algorithm practical.
- We propose a routing resource aware heuristic technique, where an RVI aware pin access allocation algorithm is proposed to improve the via insertion rate on layer Via12.
- Our detailed router outperforms an industry tool, in terms of redundant via insertion rate, wirelength, via number, and DRC violation count.

The rest of this paper is organized as follows. Section 2 introduces some preliminaries and provides the problem formulation. Section 3 gives the overview of our proposed detailed routing framework. Section 4 further describes the details of RVI enhanced detailed router, and presents RVI aware pin access allocation algorithm. Section 5 shows the experimental results, followed by conclusion in Section 6.

2. PRELIMINARIES

In this section, we briefly introduce some preliminaries on detailed routing, and then provide the problem formulation.

2.1 Graph Model

A detailed routing problem usually consists of several routing layers and each layer has a prefer direction of vertical or horizontal. Graph model is used to simplify the detailed routing problem. Given a multi-layer detailed routing problem, usually a 3-D routing graph $G = (V, E)$ representing the routing resources is constructed. On each layer, the intersection points of current layer routing grids and neighbor layers routing grids are called as *vertices*. *Edges* are routing grid segments divided by vertices. Through the routing graph, the detailed routing problem is formulated as a path finding problem. Based on the requirement of routing algorithm, the routing model could be directed or undirected. As illustrated in Fig. 2, in this paper we label the routing graph as a directed graph, and divide the whole routing problem into many subproblems.

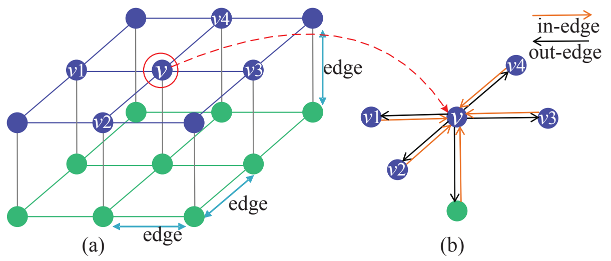

Figure 2: Graph model for detailed routing problem [15].

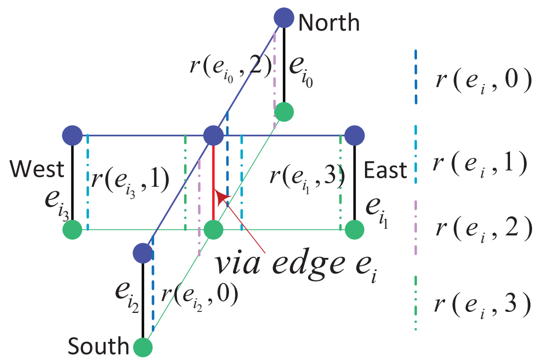

Figure 3: Direction-based redundant via variable.

2.2 Notions and Variables

The variables and sets used in this paper are presented in Table 1. The meanings and value ranges of them are also given.

Here we further introduce some new variables since the conventional variables defined in Table 1 are unable to build an effective model under the practical redundant via design rules. The position of redundant via is next to an existing via. Therefore, whether a redundant via is necessary to be inserted depends on the existence of single via. However, before a routing window is routed, the position of single via is unknown. So if we only use the existing decision variable $f(n_k, e)$ in Table 1 to control the insertion of redundant via, lots of logic constraints will be introduced to the ILP model. These logic constraints will make the problem nonlinear and more complex to solve. In order to maintain a good nature of linearity, redundant via variables related to via edges are introduced to the model.

In this work, direction-based redundant via variables are proposed as shown in Fig. 3. e_i is a via edge (without direction). Four binary variables $r(e_i, d)$ $(d = 0 \sim 3)$ related to e_i are added to ILP model. $r(e_i, 0)$, $r(e_i, 1)$, $r(e_i, 2)$ and $r(e_i, 3)$ represent the north, east, south and west redundant via variable of e_i, respectively. e_i is called as the *father edge* of these redundant via variables. If e_i is used as a redundant via, one of the four variables should be 1. Furthermore, the position of main via is also indicated by redundant via variable. For example, e_i and the neighbor via edge which is to its east compose a double via if and only if $r(e_i, 1)$ is equal to 1. At the same time, via enclosure direction is also determined.

For convenience, 0, 1, 2 and 3 represent north, east, south and west respectively when directions are represented by numbers. $opp(d)$ indicates the opposite direction of d. $opp(1) = 3$, $opp(0) = 2$, $opp(3) = 1$ and $opp(2) = 0$. In our model we introduce a variable to represent the degree of each vertex.

Definition 1 (Vertex Degree). *$dg(v_j, d)$ indicates the vertex degree of redundant via on direction d. It can be calculated by the sum of redundant via variables on direction d whose father edge terminates at v_j $(v_j \in V,\ d = (0, 1, 2, 3))$.*

2.3 Problem Formulation

Based on the notations in preceding section, we define the RVI enhanced detailed routing problem as follows:

Problem 1 (RVI enhanced detailed routing). *Given global routing (GR) and track assignment (TA) results, as well as routing region, RVI enhanced detailed routing is to route all nets in given region simultaneously. The objective is to insert as many redundant vias as possible without introducing DRC violations.*

3. OVERALL FLOW

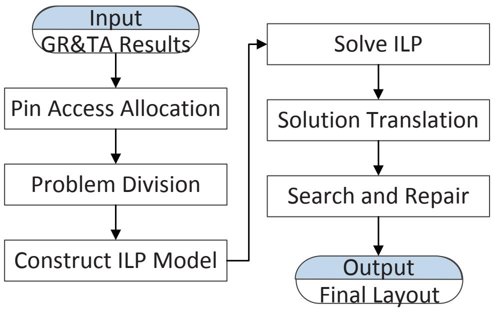

Figure 4: Overall flow of RVI enhanced detailed router

The overall flow of the proposed detailed router is illustrated in Fig. 4. The inputs of the proposed router consist of global routing results and track assignment results. Pin access allocation is the first step that will be detailed discussed in Section 4.5. Then the whole routing region will be divided into many subregions with smaller scale by global routing cell. For each subregion, detailed routing problem is formulated into an ILP problem. After solving a set of ILP problems, the 0-1 solutions will be translated into routing solution. Finally, a search and repair stage is implemented to handle the DRC violations between different regions. The output is a final detailed routing result.

4. RVI ENHANCED DETAILED ROUTER

In this section, we formulate the RVI enhanced detailed routing problem as an ILP problem based on the multi-commodity flow model. The objective function as in Eqn. (1) aims to select an optimal detailed routing and redundant via solution, while the costs are minimized. Our costs consist of via edge cost, metal edge cost, and some penalty terms about DRC violations and redundant via insertion:

$$Cost = \sum_{k=1}^{|N|} \sum_{e \in E} (f(n_k, e) \cdot c(e)). \tag{1}$$

4.1 Connectivity Constraints

The basic goal of detailed routing is that all components of each net are connected. This is guaranteed by connectivity constraints in ILP model. Connectivity constraints can be formulated as Eqn. (2), which is based on flow conservation theory.

$$\sum_{e \in E_{v_j,out}} f(n_k, e) - \sum_{e \in E_{v_j,in}} f(n_k, e) = d(n_k, v_j), \tag{2}$$

where $v_j \in V$ and $n_k \in N$.

4.2 Direction-based Capacity Constraints

In detailed routing, a *short* violation means the crossover of different nets. No short violation is the primary task of a detailed router. In routing graph, the crossover may occur on graph edges or vertices. The ILP model needs to build edge and vertex capacity constraints to remove short violation from the solution space. The work of [15] has proven that edge capacity constraints are redundant. It can be guaranteed by connectivity constraint (2) and vertex capacity constraint (3):

$$\sum_{k=1}^{|N|} \sum_{e_i \in E_{v_j}} f(n_k, e_i) \le 2. \tag{3}$$

Constraint (3) is not suitable in this work because it dose not consider redundant via variables. However, it can make help when we construct new constraints. For convenience, vertex v_j is taken as an example to present the vertex capacity constraints.

A vertex v_j is used by the routing path and it could be indicated by $\sum\limits_{k=1}^{|N|} \sum\limits_{e_i \in v_j} f(n_k, e_i) = 2$. If it is also served as a redundant via, there must be $dg(v_j, 1) = 1$. And we can find that if $dg(v_j, 1) = 1$, there must be a segment between vertex v_j and one of its neighbor vertex. Based on this observation and the introduction method of redundant via variable, in this paper we construct four direction-based capacity constraints for each vertex as Eqn. (4).

$$dg(v_j, d) + \sum_{k=1}^{|N|} \sum_{e_i \in E^d_{v_j}} f(n_k, e_i) \le 2 \quad (d = 0 \sim 3), \tag{4}$$

where for each d, $E^d_{v_j} = E_{v_j} \backslash \{(v_j, v_{j_d}), (v_{j_d}, v_j)\}$.

The new constraint of given direction has two differences compared with Eqn. (3). The first one is that the adjacent net edges of given direction are removed from Eqn. (3). The second one is redundant via variables related to given direction are added to Eqn. (3). The correctness of this constraint can be proven by enumeration method. Due to the page limit it is omitted here.

4.3 Redundant Via Insertion Method

The constraints formulated in this section aim to insert as many redundant vias as possible without introducing DRC violations.

Normally, each via edge has four neighbor via edges on via layer. As shown in Fig. 3, variables $r(e_{i_d}, opp(d))$ $(d = 0, 1, 2, 3)$ are related with edge e_i.

A. Ideal Solution. The ideal solution of redundant via insertion is achieving a 100% insertion rate. Each via has its own redundant via. In our model, the usage of via edge e_i is expressed by $\sum_{k=1}^{|N|} f(n_k, e_i)$. The capacity constraint restricts the sum to be either 0 or 1. The sum is 0 means edge e_i is unused, otherwise it is used. The generation of redundant via is expressed by $\sum_{d=0}^{3} r(e_{i_d}, opp(d))$. So constraint (5) could achieve the ideal redundant via insertion solution:

$$\sum_{k=1}^{|N|} f(n_k, e_i) = \sum_{d=0}^{3} r(e_{i_d}, opp(d)). \tag{5}$$

B. Realistic Solution. However, not all the solutions are ideal. If the insertion of a redundant via violates with around geometries, the redundant via should be given up to avoid DRC violations generation. Under this circumstance, the right item of Eqn. (5) should be 0 even the left one is 1. And Eqn. (5) is no longer suitable.

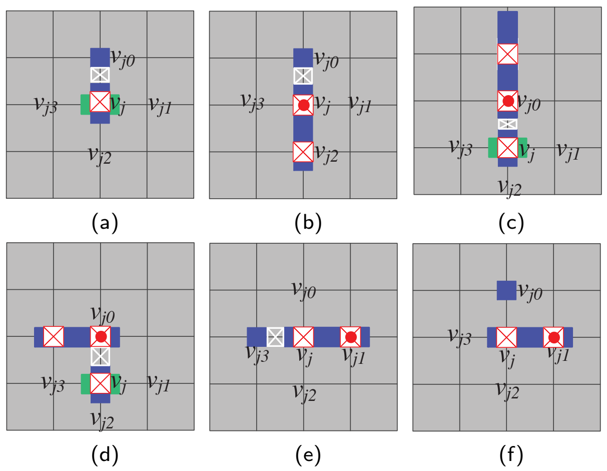

Figure 5: Layout cases for spacing constraint.

In this work, a penalty variable is added into Eqn. (5) to make the redundant via insertion realistic as Eqn. (6) shows.

$$\sum_{k=1}^{|N|} f(n_k, e_i) = \sum_{d=0}^{3} r(e_{i_d}, opp(d)) + \lambda, \tag{6}$$

where λ is a penalty variable whose value range is 0 or 1. Constraint (6) allows abandon redundant via insertion for e_i by assigning λ as 1. λ multiplied with a penalty factor μ is added to the objective function Eqn. (1) to penalize the abandon of redundant via. In our experiments, the penalty factor μ is smaller than via cost.

4.4 Via Enclosure aware Spacing Constraints

Spacing constraints considering redundant via variables and via enclosure are constructed in this subsection. Spacing rule is very important in technology file which specifies the distance between two geometries. Inequation (7) is a spacing constraint formulated in [15] for vertex pair (v_{j_1}, v_{j_2}) whose distance satisfies Eqn. (8):

$$\sum_{k=1}^{|N|}\Big(\sum_{e_i \in E^{\eta}_{v_{j_1}}} f(n_k, e_i) + \sum_{e_i \in E'_{v_{j_2}}} f(n_k, e_i)\Big) \leq 2, \tag{7}$$

$$\alpha_m + \gamma_m \leq \Gamma(v_{j_1}, v_{j_2}) < \alpha_m + \beta + \gamma_m. \tag{8}$$

Here, $\Gamma(v_{j_1}, v_{j_2})$ represents the distance of vertex v_{j_1} and vertex v_{j_2}. α_m, β and γ_m are the metal wire width, the via enclosure length, and the metal layer spacing, respectively. $E^{\eta}_{v_{j_1}}$ denotes the set of via edges of vertex v_{j_1}. $E'_{v_{j_2}}$ is the adjacent edges set of vertex v_{j_2} excluding inner edges of vertex pair (v_{j_1}, v_{j_2}), *i.e.* $E'_{v_{j_2}} = E_{v_{j_2}} \backslash \{(v_{j_1}, v_{j_2}), (v_{j_2}, v_{j_1})\}$.

Constraint (7) indicates that if the via edges of v_{j_1} is occupied by net n_k, all adjacent edges of vertex v_{j_2} can not be used by other nets again except for net n_k. However, constraint (7) is no longer capable to avoid spacing violations in this work, because it does not take redundant vias into consideration. Actually the enclosure of double via makes the spacing violation avoidance much difficult.

Without loss of generality, vertex pair (v_j, v_{j_d}) whose distance satisfies Eqn. (8) is taken as an example. New constraints are constructed by discussing the usage of via edges on vertex v_j and all edges on neighbor vertex $v_{j_d}(d = 0, 1, 2, 3)$.

Taking a vertical layer for example, the discussion is made from the following aspects.

Case 1: Vertex v_j is occupied by a single via.

In this case, Fig. 5(a) shows the geometry shape of vertex v_j on metal layer. This is the same as the problem discussed in [15]. So constraint (7) still works.

Case 2: Vertex v_j is a redundant via for neighbor vertex on vertical direction.

Here, we assume that double via consists of v_j and the south neighbor vertex, *i.e.* v_{j_2}. As shown in Fig. 5(b), v_{j_2} is a main via and v_j is a redundant via. It is obviously that $dg(v_j, 2) = 1$ represents the usage of via edge on vertex v_j. In this case, the usage of vertex v_{j_0} should be restricted. Constraint (7) is unable to avoid the spacing violation in this case. So new constraint (9) is needed to prohibit the concurrently usage of vertex v_j and vertex v_{j_0}:

$$dg(v_j, 2) + \sum_{k=1}^{|N|} \sum_{e_i \in E'_{v_{j_0}}} f(n_k, e_i) \leq 2. \tag{9}$$

In detailed routing problem, if a via edge is occupied by one net, it can be on longer used as a redundant via for neighbor vias, *i.e.* it is not possible that $\sum_{k=1}^{|N|} \sum_{e_i \in E^{\eta}_{v_j}} f(n_k, e_i) \geq 1$, meanwhile $dg(v_j, 2) = 1$. So Eqn. (7) and Eqn. (9) can be merged together.

Case 3: Vertex v_{j_0} is a redundant via, but not for vertex v_j.

In this case, we can get the following two conclusions. The first one is that $dg(v_{j_0}, 2)$ must be 0, and the second one is $dg(v_{j_0}, 0) + dg(v_{j_0}, 1) + dg(v_{j_0}, 3) = 1$ as illustrated in Fig. 5(c) ($dg(v_{j_0}, 0) = 1$) and Fig. 5(d) ($dg(v_{j_0}, 3) = 1$). From the layout in Fig. 5(c)(d), It can be found that if the usage of vertex v_j belongs to *case 1* or *case 2*, spacing violation is introduced to the routing solution. In our model, constraint (10) is formulated to eliminate this routing solution.

$$dg(v_j, 2) + \sum_{k=1}^{|N|} \sum_{e_i \in E^{\eta}_{v_j}} f(n_k, e_i) + \sum_{d=0, d!=2}^{3} dg(v_{j_0}, d) \leq 1. \tag{10}$$

Case 4: Vertex v_j and one of its neighbor vertex on horizontal direction construct a double via.

Without loss of generality, we take the east vertex v_{j_1} for example as shown in Fig. 5(e)(f). When only single via is considered, almost all the enclosure direction is preferred. But there are plenty of non-preferred direction enclosures when double via is considered. So it is necessary to build special constraints for them.

When referring to non-preferred direction via enclosures, two differences must be emphasized.

(I) The usage of edges adjacent to vertex v_{j_3} is restricted by via enclosure. As illustrated in Fig. 5(e), the via enclosure extends to the west. It is the same as *case* 1 described above but in a horizontal layer. A constraint similar to Eqn. (7) is needed to avoid spacing between vertex v_j and vertex v_{j_3}. No matter the redundant via locates at v_j or v_{j_1}, a layout like Fig. 5(e) is generated, so the new constraint is formulated as Eqn. (11):

$$dg(v_j, 1) + dg(v_{j_1}, 3) + \sum_{k=1}^{|N|} \sum_{e_i \in E'_{v_{j_3}}} f(n_k, e_i) \leq 2. \tag{11}$$

(II) The usage of adjacent vertical vertices *i.e.* v_{j_0} and v_{j_2} is no longer limited by via edges on vertex v_j. As we can see from Fig. 5(f), the extension on vertical direction of vertex v_j is half of wire width, and the enclosure influence on vertices v_{j_0} and v_{j_2} no longer exists. In other words, v_{j_1} serves as redundant via of v_j counteracts the effects of via enclosure on preferred direction neighbor vertices. Based on

Algorithm 1 RVI aware Pin Access Allocation Algorithm

```
Input: Pins and blockages;
Output: Via candidates;
1: $C \leftarrow$ searchCandidates();
2: for each candidate in C do
3:     $SgVias \leftarrow$ createSingleVias();
4:     for each via in $SgVias$ do
5:         $valid \leftarrow$ checkDRCAndAssignCost();
6:         if $valid$ = true then
7:             $RdntVias \leftarrow$ createRdntVias();
8:             for each redundant via in $RdntVias$ do
9:                 checkDRCAndUpdateCost();
10:            end for
11:        end if
12:    end for
13: end for
```

this observation, item $-dg(v_{j_1}, 3)$ is added to the left hand expression of Eqn. (7) and Eqn. (9).

If vertex v_{j_3} is served as a redundant via, the layout is similar to Fig. 5(e). Moreover, it can easily prove that east redundant via variables of v_{j_3} and west redundant via variables of v_{j_1} are impossible to be 1 at the same time. So item $-dg(v_{j_3}, 1)$ should be further added to the left hand expression of Eqn. (7) and Eqn. (9). As discussed above, Eqn. (7) and Eqn. (9) can be merged together, so we can get the constraint (12).

$$dg(v_j, 2) + \sum_{k=1}^{|N|} \sum_{e_i \in E^{\eta}_{v_j}} f(n_k, e_i) + \sum_{e_i \in E'_{v_{j0}}} f(n_k, e_i) - dg(v_{j_1}, 3) - dg(v_{j_3}, 1) \leq 2. \tag{12}$$

Likewise, Eqn. (10) also needs to be modified as in Eqn. (13) to deal with non-preferred direction via enclosure.

$$dg(v_j, 2) + \sum_{k=1}^{|N|} \sum_{e_i \in E^{\eta}_{v_j}} f(n_k, e_i) + \sum_{d=0, d!=2}^{3} dg(v_{j_0}, d) - dg(v_{j_1}, 3) - dg(v_{j_3}, 1) \leq 1. \tag{13}$$

As a result, three types of spacing constraints similar to constraints (11), (12) and (13) are added to ILP model to avoid spacing violations.

4.5 RVI Aware Pin Access Allocation

By now, we have described the ILP model in details that processes all nets concurrently, meanwhile considers redundant via insertion properly.

We make a survey on the via count on each via layer, and the statistic data shows that about 50% vias are generated on the Via12 which locates between Metal1 and Metal2. The high percent is as expected because most of the pins on Metal1 need to be lead to Metal2 by Via12. Points which are covered by pin and can be served as the candidates for via creation are called *pin accesses*. Pin access allocation is standard cell-based and proceeded pin by pin before routing stage. In this work, an RVI aware pin access allocation algorithm is proposed to increase the number of via candidates on Via12.

The details of our pin access allocation are in Algorithm 1. The inputs of pin access allocation are pins and blockages in the design, while the output is the candidate via position with creation cost. The first step is searching for candidate points which is the intersection points of routing grid covered by this pin (line 1). Taking each candidate point as center, two virtual vias with different via enclosure (preferred and non-preferred direction on Metal1) will be created (line 3). Then via creation cost is assigned to each via based on the DRC result (line 5) by the following principle: If virtual via is violated with existing geometries, *valid* in line 5 is set to false, *i.e.* the candidate point is given up; otherwise the cost is less than or equal to via cost.

For each valid virtual via, two redundant vias are inserted in turn based on enclosure direction (line 7). For example, if enclosure direction is horizontal, redundant vias are inserted on the east and west point. Then DRC procedure is executed to determine whether the redundant via is valid (line 9). If yes, the candidate cost will be further reduced.

5. EXPERIMENTAL RESULTS

The proposed algorithm is implemented with C++ language on Linux server with 2.4GHz Intel Xeon CPU and 24GB memory. It has been integrated into an industrial routing flow. In the experiments, a multi-thread algorithm is implemented utilizing the dependence between any two routing regions. It has been tested that the multi-thread algorithm can achieve a linear speedup ratio. By default, MCFRoute 2.0 utilizes 16 threads. GUROBI [16] is applied as our ILP solver.

Our experiments are executed on two benchmark suites. The first one includes eight benchmarks mapped to a 65*nm* design library. The second one includes five benchmarks mapped to a 45*nm* design library. All the benchmarks are in OpenAccess (OA) format. All engines work on and exchange data through the OA database.

Even though there are many previous published works, few of them consider design rule and report DRC results. To better demonstrate the effectiveness of our router, three RVI aware routing engines are implemented in our experiments. 1) 'MCFRoute 2.0' which is proposed in this paper has a concurrent manner both of routing and redundant via insertion. 2) 'MCFRoute + RVI' routes all net concurrently by MCFRoute [15] first and then inserts redundant via in post-routing stage by Encounter v10.10 [17]. 3) 'Encounter' processes RVI aware detailed routing by Encounter v10.10 [17], which is executed in NanoRoute mode and RVI is considered during routing stage rather than post-routing stage. In the experiments, we try to let Encounter run longer until the improvement is negligible.

The results for comparison in the experiments are wirelength ('WL') measured by *millimeter*, total via count ('#Via', $\times 10^3$), redundant via count ('#Rt', $\times 10^3$), redundant via insertion rate ('Rate'), DRC violation count ('#DRC') and runtime ('Time'). The runtime is measured by *second*.

Table 2 shows the experimental results on eight 65*nm* benchmarks. All of the three routing engines can achieve a DRC-clean results, so DRC violation count column is omitted. In Table 2, the first two columns are the name and net count ('#Net') of each benchmark. Firstly, a comparison is made between 'MCFRoute + RVI' (columns 3-7) and 'MCFRoute 2.0' (columns 13-17). Experiments of these benchmarks show that MCFRoute 2.0 can improve the redundant via insertion rate by 25.9% compared with MCFRoute with post-routing redundant via insertion by Encounter. It demonstrates that although both routing engines route nets in concurrent manner, inserting redundant via during routing stage could achieve higher insertion rate than in post-routing stage. At the same time, the wirelength is increased by 1%, but via count is reduced by 3.4%, respectively. Runtime is 1.33× of 'MCFRoute + RVI'. The experimental data is as expected because our router prefers to make detours rather than use single via if redundant via insertion is

Table 2: Results on Academic Benchmarks

benchmark		MCFRoute [15] + RVI					Encounter					MCFRoute 2.0				
Name	#Net	WL	#Via	#Rt	Rate	Time	WL	#Via	#Rt	Rate	Time	WL	#Via	#Rt	Rate	Time
b20	9309	120	61	34	55.7%	160	126	61	37	61.5%	60	121	59	41	70.7%	215
b21	9575	129	63	34	53.8%	213	133	63	38	61.2%	62	129	61	43	70.3%	282
b22	10046	177	68	42	61.8%	174	186	67	47	68.9%	65	179	65	46	70.8%	235
dma	12475	303	102	52	50.7%	301	314	102	59	57.3%	99	304	97	62	64.4%	406
frisc	15869	292	117	57	48.7%	332	304	117	69	58.8%	112	294	112	72	63.7%	429
dsp	24129	550	196	93	47.2%	735	572	198	106	53.5%	191	550	187	115	61.5%	926
pci	30835	328	158	96	61.1%	345	339	159	115	72.8%	152	329	158	125	79.3%	481
eth	43169	1006	332	183	55.2%	1025	1049	332	207	62.2%	380	1009	319	211	66.1%	1395
ratio		0.99	1.04	0.82	0.79	0.75	1.04	1.04	0.95	0.91	0.25	1	1	1	1	1

Table 3: Results on Industry Benchmarks

benchmark		Encounter						MCFRoute 2.0					
Name	#Net	WL	#Via	#Rdnt	Rate	DRC	Time	WL	#Via	#Rdnt	Rate	DRC	Time
chip1	37584	684	400311	162436	40.6%	115	1534	628	369980	221666	59.9%	72	2392
chip2	53400	832	462701	250055	54.0%	159	1478	808	401074	260785	65.0%	33	2003
chip3	101817	1696	908599	514129	56.6%	101	1849	1639	779529	506149	64.9%	54	3660
chip4	92046	1550	828688	464708	56.8%	81	1592	1504	703770	457310	65.0%	46	3357
chip5	101729	1731	907323	510295	56.2%	93	1865	1674	778938	505296	64.9%	79	3732
ratio		1.04	1.15	0.98	0.83	1.89	0.55	1	1	1	1	1	1

not valid. Secondly, we compare 'MCFRoute 2.0' with 'Encounter' (columns 8-12). Experimental results show that the proposed router improves the redundant via insertion rate by 10% compared with Encounter. It indicates that although both routing engines consider redundant via insertion during routing stage, concurrent manner could achieve higher insertion rate than sequential manner. Benefiting from concurrent routing manner of the proposed router, the wirelength, via count are also reduced by 3.6% and 3.6%, respectively. Running time is 4.06× of 'Encounter'.

'MCFRoute + RVI', 'Encounter' and 'MCFRoute 2.0' can achieve DRC-clean results on these 65nm benchmarks. In order to further verify the advantages of our concurrent router in DRC violation handling, we derive some difficult industry benchmarks. Because proceeding redundant via insertion in post-routing stage by Encounter requires a DRC-clean layout, the comparison is only made between 'Encounter' and 'MCFRoute 2.0' as shown in Table 3. The first two columns are benchmark information including name and net count. Columns 3-8 are results of 'Encounter' and columns 9-14 are results of 'MCFRoute 2.0'. From the experimental results, it can be found that the proposed router improves the insertion rate by 21% and reduces the wirelength and via count by 4% and 14% respectively compared with Encounter. Moreover, it is worth to emphasize that the router can reduce the DRC violation count by 47% for these complex industry benchmarks. The runtime is 1.82× of 'Encounter'. Both two engines use the same netlists and design rules. The better results of our router mainly profit from the concurrent manner of routing and redundant via insertion that overcomes net-order problem. The experimental results demonstrate that the proposed algorithm is capable to improve redundant via insertion rate and handle difficult routing problems while maintaining a good quality in wirelength and via count.

6. CONCLUSION

In this paper, we present an RVI enhanced concurrent detailed router based on multi-commodity flow method. The router can route all nets simultaneously, at the same time trying to insert as many redundant vias as possible. The effects of via enclosure are taken into consideration thoroughly in this paper that makes the algorithm much practical. An RVI aware pin access allocation flow is proposed to improve the redundant via insertion rate on Via12. Experimental results show the effectiveness of the proposed method. The further work of this article will focus on improving the capability of handle design rule violations in difficult benchmarks and saving our algorithm with the more challenging constraints in 22nm and beyond.

Acknowledgment

This work is supported in part by NSFC No.61274031 and CUHK Direct Grant for Research.

7. REFERENCES

[1] S.-H. Chen, K.-C. Chu, J.-Y. Lin, and C.-H. Tsai, "DFM/DFY practices during physical designs for timing, signal integrity, and power," in *ASPDAC*, 2007, pp. 232–237.

[2] K.-Y. Lee and T.-C. Wang, "Post-routing redundant via insertion for yield/reliability improvement," in *ASPDAC*, 2006, pp. 303–308.

[3] K.-Y. Lee, T.-C. Wang, and K.-Y. Chao, "Post-routing redundant via insertion and line end extension with via density consideration," in *ICCAD*, 2006, pp. 633–640.

[4] K.-Y. Lee, C.-K. Koh, T.-C. Wang, and K.-Y. Chao, "Optimal post-routing redundant via insertion," in *ISPD*, 2008, pp. 111–117.

[5] C.-K. Lei, P.-Y. Chiang, and Y.-M. Lee, "Post-routing redundant via insertion with wire spreading capability," in *ASPDAC*, 2009, pp. 468–473.

[6] S.-T. Lin, K.-Y. Lee, T.-C. Wang, C.-K. Koh, and K.-Y. Chao, "Simultaneous redundant via insertion and line end extension for yield optimization," in *ASPDAC*, 2011, pp. 633–638.

[7] J. Pak, B. Yu, and D. Z. Pan, "Electromigration-aware redundant via insertion," in *ASPDAC*, 2015, pp. 544–549.

[8] G. Xu, L.-D. Huang, D. Z. Pan, and M. D. Wong, "Redundant-via enhanced maze routing for yield improvement," in *ASPDAC*, 2005, pp. 1148–1151.

[9] H. Yao, Y. Cai, X. Hong, and Q. Zhou, "Improved multilevel routing with redundant via placement for yield and reliability," in *GLSVLSI*, 2005, pp. 143–146.

[10] H.-Y. Chen, M.-F. Chiang, Y.-W. Chang, L. Chen, and B. Han, "Full-chip routing considering double-via insertion," *IEEE Transactions on CAD*, vol. 27, no. 5, pp. 844–857, 2008.

[11] C.-T. Lin, Y.-H. Lin, G.-C. Su, and Y.-L. Li, "Dead via minimization by simultaneous routing and redundant via insertion," in *ASPDAC*, 2010, pp. 657–662.

[12] "LEF/DEF language reference," http://www.ispd.cc/contests/14/web/doc/lefdefref.pdf.

[13] C. Albrecht, "Provably good global routing by a new approximation algorithm for multicommodity flow," in *ISPD*, 2000, pp. 19–25.

[14] T.-H. Wu, A. Davoodi, and J. T. Linderoth, "GRIP: scalable 3D global routing using integer programming," in *DAC*, 2009, pp. 320–325.

[15] X. Jia, Y. Cai, Q. Zhou, G. Chen, Z. Li, and Z. Li, "MCFRoute: a detailed router based on multi-commodity flow method," in *ICCAD*, 2014, pp. 397–404.

[16] Gurobi Optimization Inc., "Gurobi optimizer reference manual," http://www.gurobi.com, 2014.

[17] "Cadence SOC Encounter," http://www.cadence.com.