# Redundant-Via Enhanced Maze Routing for Yield Improvement


Gang Xu
CS Department
University of Texas at Austin
Austin, TX 78712, USA
xugang@cs.utexas.edu
Tel: 1-512-471-9588

Li-Da Huang
Texas Instruments
Austin, TX 78746, USA
lida@ti.com
Tel: 1-512-732-8308

David Z. Pan
ECE Department
University of Texas at Austin
Austin, TX 78712
dpan@ece.utexas.edu
Tel: 1-512-471-1436
Fax: 1-512-471-8967

Martin D.F. Wong
ECE Department
University of Illinois at Urbana Champaign
Urbana, IL 61801, USA
mdfwong@uiuc.edu
Tel: 1-217-244-1729



**Abstract - Redundant via insertion is a good solution to reduce the yield loss by via failure. However, the existing methods are all post-layout optimizations that insert redundant via after detailed routing. In this paper, we propose the first routing algorithm that considers feasibility of redundant via insertion in the detailed routing stage. Our routing problem is formulated as maze routing with redundant via constraints. The problem is transformed to a multiple constraint shortest path problem, and solved by Lagrangian relaxation technique. Experimental results show that our algorithm can find routing layout with much higher rate of redundant via than conventional maze routing.**


## I. Introduction

As VLSI feature size continues to shrink to deep sub-micron (DSM) regime, it has become a heavy burden for VLSI manufacturers to maintain good manufacturability and high yield. Manufacturability must be considered in the design flow. A new research topic, known as design for manufacturability (DFM), has become active recently [1][2].

Among DFM problems, how to reduce yield loss by via failure is one of the most important. Vias are components in VLSI circuits to connect wire segments on different metal layers. Vias may fail partially or completely in manufacturing process. This is particularly true in DSM process due to various reasons such as cut misalignment, electro migration, and thermal stress induced voiding. A complete via failure will result in a broken net; a partial via failure will increase the resistance of the signal net and lead to delay penalty and timing problems. Yield loss by via failure thus becomes critical and requires a good control.

A good solution to reduce yield loss by via failure is to add a redundant via adjacent to each single normal via as a backup (Fig 1). Here we refer to single normal vias as vias on the wire with minimal wire width. In the rest of the paper a via usually refers to a single normal via. Unlike multiple vias on the wide wire, redundant vias are not required in functionality. However, with redundant vias a net is less likely to break. Redundant vias also decrease the resistance of via and alleviate the delay penalty by partial via failures.


This work was partially supported by the National Science Foundation under grant CCR-0306244 and IBM Faculty Award. We used computers donated by Intel Corporation.


In fact, redundant via insertion has been strongly recommended by major foundries in their 130nm and 90nm processes [3] to improve the yield. Major EDA vendors such as Cadence and Synopsis have already added the feature of redundant via insertion to their latest routers (Cadence Nanoroute, Synopsis Astro). There are also third-party EDA tools such as Nannor Acuma and Prediction EYE/PEYE [4] specially designed to insert redundant vias.

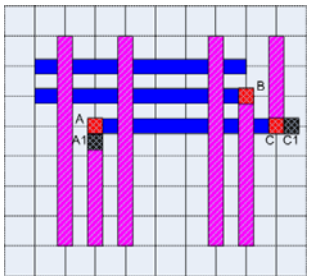


**Fig 1 Redundant vias.** A1 and C1 are redundant vias of A and C respectively. We are unable to insert the redundant via for B because of the minimum spacing rule.

However, all these tools are doing redundant via insertion in the post-layout stage following detailed routing. That is, redundant vias are inserted after the layout is almost determined. Because at this stage only the slight layout modification is allowed, this methodology will restrict the feasibility of redundant via insertion. A better idea is to consider the redundant via insertion in the routing stage, which has been foreseen as one of the future routing challenges in DSM and nanometer era [5].

In this paper, we propose a maze routing algorithm that considers the feasibility of redundant via insertion in the routing stage. To our best knowledge, this is the first study in this direction in public domain. In our algorithm, a 2-pin net is routed with the constraint on the maximum number of dead vias in each net. Here dead vias refer to vias ineligible to have redundant vias. For example, via B in Fig 1 is a dead via. By assigning cost to each routing edge, the maze routing problem with redundant via constraints is transformed to a

multi-constraint shortest path problem, and solved by Lagrangian relaxation technique. Experimental results show that our algorithm can find routing layout with much higher rate of redundant via than conventional maze routing.

The paper is organized as follows. Section 2 presents the problem formulation; section 3 studies the solution to a special case, and then extends it to the general case. Experimental results are shown in section 4. Finally, section 5 will conclude the paper.

## II. Problem Formulation

In this paper we use the maze routing algorithm, which is a grid based sequential routing algorithm. The routing region in maze routing is represented as a k-layer grid graph. An x-axis or y-axis edge on a layer represents a wire segment. A via corresponds to a z-axis edge connecting a pair of vertices at the same x-y coordinate on the two neighboring layers. Obstacles and occupied vertices and edges are removed from the graph because they are not available as routing resource. In the following part of this paper sometimes we may refer the vertex to a via for simplicity. As a sequential routing algorithm, maze routing seeks net routes one by one in a certain pre-set order. Once a net is routed, the vertices and edges representing pins, vias, and wire segments of this net are occupied and then removed. Nets being routed late will have less routing resource. See [6] for more details about maze routing.

Some basic concepts are required to discuss redundant via constraint. For any vertex v representing a single normal via in the grid graph, we define its adjacent vertices as neighbors. The unoccupied neighbors of v are called *off-track neighbors*, and the neighbors only occupied by the net that v belongs to is called *on-track neighbors*. On-track and off-track neighbors of v are *free neighbors*. The total number of free neighbors of v is defined as the *degree of freedom (DoF)* of v. Vias with non-zero DoF are *alive*. Otherwise, they are *dead*. A via with only one free neighbor is *critical*. Fig 2 shows these concepts.

The redundant via must be inserted between v and one of its free neighbors in order to satisfy the minimum spacing rule, as shown in Fig 1. Therefore, only living vias can have redundant vias. We constrain the maximum number of dead vias in each net to guarantee redundant via insertion. The constraint is per net because we want to balance the number of dead vias in each net. Without the balance some net may have much more dead vias than others, and its performance is likely to degrade much due to partial via failures.

Assuming we are routing net m, we formulate the maze routing problem with redundant via constraints as follows:

Problem 1. Maze routing with redundant via constraints (MRRVC):

*Find the shortest route for net m such that* $\forall i::1 \le i \le m : DV_i \le C_i$ *, where* $DV_i$ *is the number of dead vias in net i,* $C_i$ *is the constraint of net i.*

## III. Problem Solution

In this section we present the solution to the MRRVC problem. First we study a special case of this problem. The problem is transformed to a multi-constraint shortest path problem, and solved by Lagrangian relaxation technique.

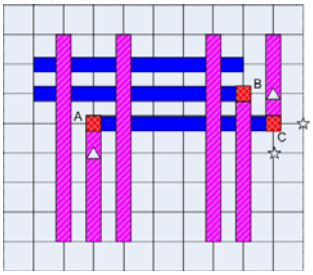


**Fig 2 Free neighbors and the degree of freedom of a via**. Stars and triangles indicate free neighbors of A and C. Stars are off-track neighbor; triangles are on-track neighbor. A is a critical via because its DoF is 1. B is a dead via. The DoF of C is 3.

Then we extend this solution to the general case.

### A. *The solution to MRRVC: a special case*

We consider the following special case: before the m-th net is routed, all vias alive are critical. In this scenario the routing layout is dense and every via is easy to be killed, as shown in Fig 3. By assigning cost to every edge in the current routing graph, we can count how many vias in the routed net i will be killed when routing a new net m.

The algorithm of cost assignment is shown in Fig 4. Initially, costs of all edges are set to zero. Then we scan free neighbors of each via alive, and increase cost of each incident edges of the free neighbor by one.

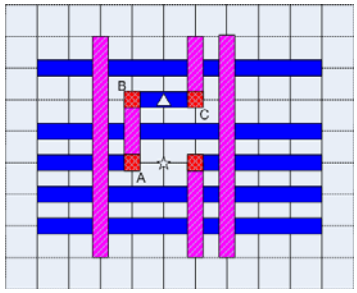


**Fig 3 A special case that all via alive are critical**.

```
Input: net i, the routing graph
Output: cost assignment
for each edge e in the routing graph
    cost(e) = 0;
for each living via v in i
    for each free neighbor n of v
        for each incident edge e of n
            cost(e) ++;
```

**Fig 4 The algorithm of edge cost assignment**

Now we have the following theorem to count killed vias.

*Theorem* 1: $\forall i::1 \le i \le m-1 : \sum_{e \in m} c_i^e = 2KV_i$ , where e represents an edge, $c_i^e$ is cost of e regarding net i, $KV_i$ is the total numbers of vias killed by net m.

Notice that routed nets can also kill vias in the new net. In this case it is easy to count these vias. We just assign cost to each z-axis edge in the current graph in the following way: Looking on the z-axis edge as a via, assign one if it is dead, and zero otherwise. Assign zero to all x-axis and y-axis

edges. Now we have the following theorem to count dead vias in the new net m.

*Theorem 2*: $\sum_{e\in m} c_m^e = DV_m$, where e represents an edge, $c_m^e$ is cost of e regarding net m, $DV_m$ is the total number of dead vias of net m.

To count dead vias in each net, we assign a cost vector $(c_1^e, c_2^e, ..., c_{m-1}^e, c_m^e)$ to each edge in the current graph, where $c_1^e, c_2^e, ..., c_{m-1}^e$ are assigned in the way of theorem 1, and $c_m^e$ is assigned based on theorem 2.

Obviously, MRRVC is equivalent to the following MCSP problem.

Problem 2: Multi-constrained shorted path (MCSP)

*Given a graph where each edge is assigned a cost vector* $(c_1^e, c_2^e, ..., c_{m-1}^e, c_m^e)$, *two vertices s and t in the graph, and a constraint vector* $(C'_1, C'_2, ..., C'_{m-1}, C'_m)$, *find a shortest path P from s to t such that* $\forall i :: 1 \le i \le m : \sum_{e\in P} c_i^e \le C'_i$, *where* $C'_i$ *is twice of the number of vias in net i killed by net m, i<m, and* $C'_m$ *is the number of dead vias in net m.*

The MCSP problem was studied in [7] and [8]. It is proved to be NP-hard. But we can get a heuristic solution based on Lagrangian relaxation technique. Given a MCSP problem instance and a non-negative constant vector $(\lambda_1, \lambda_2, ..., \lambda_{m-1}, \lambda_m)$, in which each element represents the weight of a constraint, we can construct the following unconstrained Lagrangian sub-problem.

Problem 3: Lagrangian Sub-Problem (LSP)

*Minimize* $\sum_{e\in P} 1 + \sum_{i=1}^{m} \lambda_i (\sum_{e\in P} c_i^e - C'_i)$.

Equivalently, it is:

*Minimize* $\sum_{e\in P} (1 + \sum_{i=1}^{m} \lambda_i c_i^e) - \sum_{i=1}^{m} \lambda_i C'_i$

Because the last term is constant, the above LSP can be solved optimally by the weighted shortest path algorithm in polynomial time.

The element in the vector of constraint weight is known as Lagrangian multiplier. Let the multiplier be variable, the Lagrangian multiplier problem (LMP) for MCSP is defined as follows.

Problem 4: Lagrangian Multiplier Problem (LMP).

*Maximize* $L(\lambda_1, \lambda_2, ..., \lambda_{m-1}, \lambda_m)$

*subject to* $\forall i :: 1 \le i \le m : \lambda_i \ge 0$ .

The nice property of LMP is that the optimal solution is the lower bound of the optimal solution of MCSP because of the following inequality:

For any $(\lambda_1, \lambda_2, ..., \lambda_{m-1}, \lambda_m)$, $\lambda_i \ge 0$,

$$\min_P \{\sum_{e\in P} 1 : \forall i :: 1 \le i \le m : \sum_{e\in P} c_i^e \le C'_i\}$$

$$\ge \min_P \{\sum_{e\in P} 1 + \sum_{i=1}^{m} \lambda_i (\sum_{e\in P} c_i^e - C'_i) : \forall i :: 1 \le i \le m : \sum_{e\in P} c_i^e \le C'_i\}$$

$$\ge \min_P \{\sum_{e\in P} 1 + \sum_{i=1}^{m} \lambda_i (\sum_{e\in P} c_i^e - C'_i)\} = L(\lambda_1, \lambda_2, ..., \lambda_{m-1}, \lambda_m) .$$

In addition, LMP is a convex programming problem, and thus can be solved by non-linear programming techniques, like the sub-gradient method. The lower bound theorem and the convexity of LMP inspire us to use the solution to LMP as an approximation of the solution to MCSP.

### B. *The solution to MRRVC: the general case*

Now we need to consider the general case where there may exist non-critical vias, i.e., vias with more than one free neighbors. If we just use the algorithm of cost assignment in Fig 5, as shown in Fig 5, we have the following theorem for the general case.

*Theorem 3*: $\sum_{e\in m} c_i^e = 2DoFL_i$, where e represents an edge, $c_i^e$ is cost of e regarding net i, $DoFL_i$ is the total numbers of DoF loss of all vias in net i due to addition of net m.

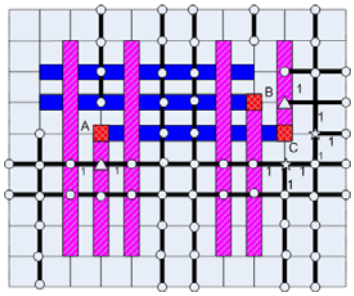


**Fig 5 Cost assignment in the general case**. Vertices and edges in the current graph are provided for clarity. Incident edges of all free neighbors of C are assigned cost. Sum of DoF loss cannot indicate the number of dead vias in the net that via A and C belong to. A new net passing via A and C's free neighbors will lead to DoF loss 6. But only A is killed.

Different from the special case, sum of DoF loss in the general case does not exactly indicate the number of vias being killed. It is just a heuristic. In addition, the sum of DoF loss is not very accurate to predict the number of dead vias, especially when DoF of vias greatly vary. For example, killing a living via with four DoF and killing four critical vias both lead to the same DoF loss. Although the latter one is strongly undesired, the router cannot distinguish it by the DoF loss.

An improved cost assignment is to assign the weighted edge cost based on the DoF: if the DoF of a via is *x*, assign *1/x* to edges incident to the free neighbor, as shown in Fig 6.

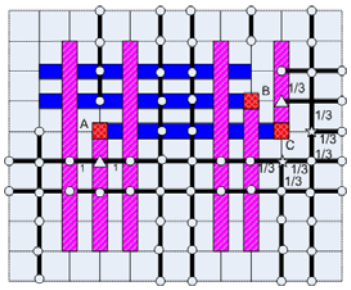


**Fig 6 Improved cost assignment in the general case.** A new net passing via A and C's free neighbors will lead to weighted DoF loss 10/3 .

In this case, we have the following theorem:

*Theorem 4*: $2KV_i \le \sum_{e\in m} c_i^e \le 2DoFL_i$, where e represents an edge, $c_m^e$ is cost of e regarding net i, $KV_i$ is the total

numbers of vias being killed by net m, $DoFL_i$ is the total numbers of DoF loss due to addition of net m.

Theorem 4 shows that the improved cost assignment is a better upper bound of dead vias. However, this heuristic still overestimates the number of dead vias. To compensate the overestimation, we can relax the constraints in MCSP problem. For example, we can record the average overestimated number of dead vias in previous routing steps as a reference.

The scheme of cost assignment can be further improved if we notice a good property of some vias: *safety*. A free neighbor n of a via v is a *safe neighbor* if the degree of n is no more than 1, i.e., n is an isolated or dangling vertex. A via is *safe* if it has a safe neighbor. For example, in Fig 5 and 6 via C is safe because the triangle neighbor above C only has one incident edge. For a safe neighbor we have the following theorem:

*Theorem 5*: A safe neighbor will never be occupied by a new net.

Therefore, if a via v has a safe neighbor n, we can always insert redundant via in edge (v, n). The via will never be killed. So we can just ignore these vias when assigning cost. For example, in Fig 10 and 11, the cost assigned to incident edge of via C's triangle free neighbor can actually be removed, because C has a safe neighbor and its redundant via is guaranteed.

## IV. Experimental results

We implement the constrained maze routing algorithm based on Lagrangian relaxation in C++. The platform is a Sun Ultra 60 workstation with 2GB memory. Our router is a multi-layer router.

We have three test cases. In case I, 200 nets are routed in a 40x40 grid; in case II, 400 nets are routed in a 120x120 grid; in case III, 800 nets are routed in a 200x200 grid. The net lists in these test cases are generated randomly. We have a 6-layer routing region. We set the constraint on the number of dead vias as D0: no dead vias allowed at all; D1: no more than 1 dead via per net; and D2: no more than 2 dead vias per net. The routing results are compared with the result of the conventional shortest path maze routing algorithm (C), as shown in Table 1. The comparison metrics include routable nets (column II), the routability, i.e., the percentage of routable nets (column III), the total number of vias (column IV), the number of dead via (column V), and the rate of redundant via insertion (column VI). The rate of redundant via insertion is calculated by the following equation: $R = 1 - DV / TV$ , where TV represents the total number of via, and DV is the total numbers of dead vias.

As we can see, our maze routing algorithm can obtain up to 200% higher rate of redundant via insertion than the conventional maze routing if no dead via is allowed, as shown in case I. By relaxing the constraint to allow no more than two dead vias per net, we can still improve the rate of dead via insertion remarkably: 25% in case II, 36% in case III, with the slight loss of routable nets 2.6% and 1.2%. As to the case I, although the 10% loss of routable nets seems not negligible, the huge 80% improvement on the rate of redundant via insertion pays off.

**Table 1. Comparison of the routing results in the 6-layer routing region.**

| Case | I | II | III | IV | V | VI |
|---|---|---|---|---|---|---|
| I | D0 | 97 | 48.5% | 426 | 0 | 100% |
| | D1 | 108 | 54% | 431 | 60 | 86.1% |
| | D2 | 142 | 71% | 521 | 195 | 62.6% |
| | C | 165 | 82.5% | 547 | 367 | 32.9% |
| II | D0 | 292 | 73% | 989 | 0 | 100% |
| | D1 | 359 | 89.75% | 1138 | 205 | 82.0% |
| | D2 | 382 | 95.5% | 1159 | 403 | 65.2% |
| | C | 392 | 98% | 1133 | 546 | 51.8% |
| III | D0 | 546 | 68.25% | 1937 | 0 | 100% |
| | D1 | 689 | 86.13% | 2401 | 385 | 84.0% |
| | D2 | 773 | 96.6% | 2504 | 849 | 66.1% |
| | C | 782 | 97.75% | 2377 | 1223 | 48.5% |

## V. Conclusions

In this paper we present a constrained maze routing algorithm that can guarantee the high rate of redundant via insertion, which is important to reduce yield loss by via failure in today's DSM VLSI manufacturing. By assigning cost to each edge, the problem of maze routing with redundant via constraints is first transformed to a multi-constrained shortest path problem, and then solved by Lagrangian relaxation technique. The experimental results show that the rate of redundant via insertion can be improved up to 200% by our approach.

## References


[1] Louis K. Scheffer, "Physical CAD Changes to Incorporate Design for Lithography and Manufacturability" *Proc of ASPDAC*, 2004.

[2] Y. Zorian, D. Gizopoulos, C. Vandenberg, P. Magarshack, "Guest Editors' Introduction: Design for Yield and Reliability", *IEEE Trans on Design & Test of Computers*, vol 21 , issue 3, May 2004.

[3] TSMC Symposium 2004.

[4] G. A. Allan and A. J. Walto, "Automated redundant via placement for increased yield and reliability", *Proc of SPIE*, vol 3216, pp 114-125, 1997.

[5] Hardy K.S. Leung, "Advanced Routing in Changing Technology Landscape", *Proc of ISPD*, pp. 118-121, 2003.

[6] Naveed Sherwani, "Algorithms for VLSI Physical Design Automation",3rd edition, *Kluwer Academic Publisher*, 1999.

[7] Hai Zhou, Martin D.F. Wong, "Crosstalk-Constrained Maze Routing Based on Lagrangian Relaxation", *Proc of ICCD*, pp. 628-633, 1997.

[8] Li-Da Huang, Martin D.F. Wong, "Optical Proximity Correction (OPC)-Friendly Routing", *Proc of DAC*, pp. 186-191, 2004.