2012

3D Global Router: a Study to Optimize Congestion, Wirelength and Via for Circuit Layout

Yue Xu
Iowa State University

3D global router: A study to
optimize congestion, wirelength and via for circuit layout

by

Yue Xu

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Major: Electrical Engineering

Program of Study Committee:
Chris Chu, Major Professor
Degang Chen
Randall L. Geiger
Akhilesh Tyagi
Zhao Zhang

Iowa State University
Ames, Iowa
2012

DEDICATION

To my parents and dear friends

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to take this opportunity to express my thanks to those who helped me on so many aspects for my research and this thesis.

First of all, Dr. Chris Chu for brining me into Iowa State University. During my Ph.D years, his kind guidance, encouragement and support helped me working on the advanced topics in VLSI CAD area. More importantly, he setup a model for me to learn from and gave so many advices that would be invaluable throughout my life.

Furthermore, I would also like to express my deep thanks to my other committee members: Dr. Degang Chen, Dr. Randall L. Geiger, Dr. Akhilesh Tyagi, Dr. Zhao Zhang for their insightful advices, despite of their busy schedules. It has been a great honor to study from and working along with them.

To my friends for their loving guidance and companionship during my study at ISU.

Finally, I would like to dedicate this thesis to my parents without whose support I would not have been able to complete this work.

ABSTRACT

The increasing size of integrated circuits and aggressive shrinking process feature size for IC manufacturing process poses significant challenges on traditional physical design problems. Various design rules significantly complicate the physical design problems and large problem size abides nothing but extremely efficient techniques. Leading physical design tools have to be powerful enough to handle complex design demands and be nimble enough to waste no runtime. This thesis studies the challenges faced by global routing problem, one of the traditional physical design problems that needs to be pushed to its new limit. This work proposes three effective tools to tackle congestion, wire and via optimization in global routing process, from three different aspects.

The number of vias generated during the global routing stage is a critical factor for the yield of integrated circuits. However, most global routers only approach the problem by charging a cost for vias in the maze routing cost function. The first work of this thesis, FastRoute 4.0 presents a global router that addresses the via number optimization problem throughout the entire global routing flow. It introduces the via aware Steiner tree generation, 3-bend routing and layer assignment with careful ordering to reduce via count. The integration of these three techniques with existing academic global routers achieves significant reduction in via count without any sacrifice in runtime.

Despite of the recent development for popular rip-up and reroute framework, the congestion elimination process remains arbitrary and requires significant tuning. Global routing has congestion elimination as the first and foremost priority and congestion issue becomes increasingly severe due to timing requirements, design for manufacturability. The second work of this thesis, an auction algorithm based pre-processing framework

(APF) for global routing focuses on how to eliminate congestion effectively. In order to achieve more consistent congestion elimination, the framework uses auction based detour techniques to alleviate the impacts of greedy sequential manner of maze routing, which remains as a major drawback in the most popular global routing framework. In the framework, APF first identifies the most congested global routing locations by an interval overflow lower bound technique. Then APF uses auction based detour algorithm to compute which nets to detour and where to detour. The framework can be applied to any global routers and would help them to achieve significant improvement in both solution quality and runtime.

The third work in this thesis combines the advantage of the two framework used to minimize via usage in global routing: 3D routers with good solution quality and efficient 2D routers with layer assignment process. It results in a new multi-level 3D global router called MGR (multi-level global router) that combines the advantage of both kinds. MGR resorts to an efficient multi-level framework to reroute nets in the congested region on the 3D grid graph. Routing on the coarsened grid graph speeds up the global router while 3D routing introduces less vias. The powerful multi-level rerouting framework wraps three innovative routing techniques together: an adaptive resource reservation technique in coarsening process, a new 3-terminal maze routing algorithm and a network flow based solution propagation method in uncoarsening process. As a result, MGR can achieve the solution quality close to 3D routers with comparable runtime of 2D routers.

CHAPTER 1. OVERVIEW

Circuit design is a fascinating field. Its products support the daily functioning of our society in almost every corner. The process of circuit design is highly standardized. Foundries provide a standard library with fundamental logic. System design depicts the behavior of circuit. Logic synthesis breaks down such behavior into basic logics and how all the elements should form as an integral to accomplish the functionality. Last but not least, physical design determines where the standard cells should be in a layout and how to use metal wires to connect cells together. This standardized design procedure is applied to most circuit design, while analog, RF and power circuit design processes rely more heavily on manual design due to their high level of variety and small problem size.

As circuit functionality becomes more intricate, the number of instances used keeps on growing. Nowadays, a logic circuit could easily contains millions of standard cells. Manually integrate them is just impossible. Electronic design automation (EDA), a specific type of computer-aided design, comes as a rescue to accomplish the complex circuit design tasks.

This thesis lies within the area of electronic design automation in physical design. More specifically, it focus on global routing to reduce congestion, wirelength and via count for a circuit, which results in better timing, higher yield.

1.1 Introduction

Traditional physical design problems, or automated layout design problems, which covers the entire process of automatic digital circuit layout generation from a given netlist, was considered well studied while the truth story is quite the contrary. Although the most fundamental problem remains the same during the past decade, physical design in circuit design area faces new challenges as day goes by. These challenges comes from the ordinary demands we place on those little circuits that millions of designers devote their daily effort on: powerful and faster circuits yet on the smallest possible area and consuming least power.

The desirable circuits pose multiple challenges for physical design. First comes the problem size. All the fancy applications the world hypes about require enhanced functionality from hardware side, which directly translates into more gates and thus more variables to decide for physical design problem. Because most of physical design problem is NP hard, runtime could easily get exploded if the underlying techniques is not runtime scalable. Then comes the speed of circuit. As the speed of CPU in a mobile devices already surpassing 1GHz and desktop CPU at 2GHz, timing requirement poses significant constraints. With the beefed up complexity and speed, consumer electronics giants would not sacrifice the battery life of their handhelds while Internet moguls refuse the idea of melting data centers. Circuit power becomes another issue to take care of in physical design. This sets more constraints and adds on more objectives to optimize in physical design. Both the functionality and speed just mentioned could not be realized without the help of shrinking process feature size. But that brings no good news for automated layout design either. With manufacturing process approaching its physical limits, IC foundries churn out more and more design rules to make sure their circuits would function properly. Sometimes it may even invalidate classical physical design solutions and requires a brand new one. The many complex design rules tend to have very

discrete nature and thus very hard to model and handle. The list of challenges could go on and on to a point of almost desperation. But on a second thought, all these challenges bring some of the brightest idea into the field of physical design automation.

Due to the increasing size of modern circuits, together with complex demands from performance and manufacturability, every major physical design automation problem faces an exploding size and intricate trade-off to balance. The sheer problem size requires efficient solution methods. Meanwhile, increasingly complex constraints imposed by timing, power and design rules requires physical design tools to spend extra effort to configure a satisfying solution, especially when those constraints intend to push solutions to different corners.

Due to different problem nature for each physical design area, some problems face much tougher challenges than the others. As the last step of physical design flow, routing is the worst hit area. Routing in physical design is the step that uses metal wire to connect standard cells or functional blocks together. Unlike the physical design steps before routing, such as floorplan and placement, routing problem could hardly be modeled as a classical numerical optimization problem. It has a very discreet nature so adding constraints complicates the solution greater than the other steps. In addition, some design features, like timing, circuit power and signal integrity directly depends on the wiring solutions. They directly sets constraints on routing solution but are merely considered as a secondary optimization parameter in earlier stages. Not only that, when timing or power goes wrong with a solution, designers are typically reluctant to go back to earlier stages, they would rather rerun routing with some extra guidance and hope it will generate desirable solution. So the best approach is the design characterization and rerouting feedback loop to optimize design features.

Routing stage typically consumes most of runtime, requires most attention and needs to handle most of the design issues and thus requires special cares. To make the matter worse, the traditional issues for routing, such as congestion and prohibited routing pat-

terns, turn out to be more difficult to handle as process technology advances. All the above mentioned factors put routing onto the center of physical design area.

Modern designs are liable to congestion problems because of the increasingly concentrated routing demands. Designs with IP blocks usually create narrow channels which further increase the difficulty of routing. Routability has become a major issue for the large designs. In addition, as the continuous shrinking feature size poses great difficulty on manufacture process. Routing is a key step to consider the design-for-manufacture/yield (DFM/DFY) during the design process. The most direct method is to handle advanced design rules, which could involve multiple objects and even cross layers. Vias, one major source for circuit failure, have larger process variation that impacts the timing/yield of circuits in a less predictable way. The number of vias has become a standard parameter to minimize in global routing. The reasons why via is so important in global routing come twofold.

Via has a higher probability to cause open connection due to its manufacturing process, which lowers circuit yields. To make it worse, via has large variations in its resistance value, which causes performance degradation. Although double via insertion and post routing optimization eases the severity of such issues, the number of vias in routing stage remains as one of the determining factors in solution quality. In addition, there are complex design rules involving vias, which typically requires extra spacing around vias. This, together with double via insertion, means more vias than necessary may consume too much routing area and lead to exacerbated congestion problem. So nowadays the standard routing problem formulation has congestion elimination, wirelength minimization and via count minimization as the primary goal. This problem could be very complicated to solve because three different objectives and intricate dependencies among the objectives.

In order to make the complexity manageable, the routing problem is usually solved by a two-stage approach: a global routing stage followed by a detailed routing one. Global

routing works on the routing area partitioned into tiles. It allocates the routing demand globally over the chip area and guides the subsequent detailed routing to finish the track assignment and via creation. On the other hand, detail routing uses actual metal shapes or vias to realize final wiring solution. Although global routing neglects the routing details such as tracks and design rule checks, it generates interconnect information very close to the final routing implementation and can be used for accurate estimation of interconnect topology, wirelength, congestion and timing.

This work mainly focuses on congestion and via count reduction techniques in global routing for the following reasons. Congestion and via count are two of the most important features in routing problem. Timing requirement and violation free condition is always harder to attain for a routing solution with congestion hot spot, comparing to a solution without it. With routing demands smoothed out, it is much easier to add wire shielding to set apart signal aggressor from victims or insert double vias to enhance yield. For vias, less of them would involve less objects that could induce design violations, especially for advanced technology in which there is a large number of complex design rules associated with via. I choose global routing because it is a more suitable research topic than the entire routing stage or detail routing. Advances in detail routing needs close collaboration with foundries and typically require at least a small team to work out a functioning solution, if not more. Furthermore, improvement global router's ability to handle congestion and reduce via count could provide a solid backbone for the other topics in routing area to advance.

Global Routing is one of the most traditional computer aided physical design problem. In global routing, the layout regions are partitioned into coarse cells. Capacity is assigned to the grid edge abstracted from the routing resources between two neighboring cells to model how much wires could pass the boundaries depending on feature size, cell size and intra-cell wire usage to limit the number of wire allocation on that boundary. Routing usage is the actual number of wires that will cross on that specific global cell boundary.

The fundamental goal for the global router is to realize all connections while minimizing total wirelength which consists of metal wirelength and via count while conforming to all resource constraints, i.e. the usage at all boundaries of global cells should be less than or equal to the corresponding capacity.

When wiring demands exceed the resources, or usage is higher than capacity in other words, routing congestion arises. Congestion is an exacerbating issue for global routing due to the fact that the growth of wiring demands keeps on out pacing the growth of wiring resources [1]. The fact that the number of metal layers continues to grow is an indirect evidence of routing resource shortage. Although we can always add more metal layers in theory, it is unwise to do so due to manufacturability and cost. To make the matter worse, modern IC designs tend to have extremely congested local regions caused by complicated on-chip communication, IP blocking or timing requirements. Global router spends the majority of runtime to detour nets involved in congested locations. Detour usually comes along with a price tag of longer wirelength. It may significantly deteriorate the timing for a design and complicate detailed routing task. Congestion elimination becomes the key feature to tell the performance of global routers due to the increasing severity of congestion.

There are two major categories of global routing frameworks: concurrent framework and sequential framework. Concurrent approach tries to handle multiple nets simultaneously. Albrecht et. al. [2] [3] proposed a multi-commodity flow approximation algorithm to solve the global routing problem. The flow technique is used to solve a linear programming relaxation of global routing. Although the idea of sing numerical optimization to solve a problem with discrete nature in concurrency is quite elegant and should work well in theory, the tool showed inferior performance comparing to sequential routers even for easy benchmarks. The fundamental cause for such inferior performance is due to the discretization step to convert the continuous multi-commodity flow to global routing solution that loses accuracy and optimality. Other concurrent global routing works all

use integer linear programming (ILP) as the underlying solver. BoxRouter [4] employed a hybrid approach with the application of ILP to simultaneously handle multiple nets and achieved shorter runtime but still falls far behind sequential routers on account of runtime. For ILP based global routers, the drawback mainly lies on the reluctance and difficulty to correctly detour. Only after numerous efforts spent by ILP fail and indicate that current set of candidate routes cannot generate congestion free solutions, will concurrent router include new candidates with more detours. Even though [5] shows significant improvement in terms of wirelength comparing to sequential routers, its excessive runtime prohibits it from practical usage. Besides, concurrent routers tend to generate results with more remaining congestions when the benchmarks are hard.

On the contrary, Sequential approach generally employs a rip-up and reroute (R&R) framework. Such sequential rip-up and reroute framework is the most successful global routing framework for congestion elimination is the foremost concern. In general, the sequential framework uses pattern routing [6] to initialize a routing solution. With the initial solution, the framework uses maze routing [7] to sequentially rip-up a single net currently using congested region and reroute the net to minimize its routing cost. Maze routing usually uses the classical shortest path algorithms like Dijkstra's algorithm [8] or A* search [9]. In this manner, sequential global routers guide nets one by one to avoid congested regions. The greedy nature lacks the global view about the entire problem and tends to get stuck at local minimum solutions quite easily. Although this approach has no guarantee to achieve optimal solution, it has been proved to be very effective in practice and is considerably faster than concurrent approach.

Depending on how they take via into consideration, there are generally two kinds of global routers. 3D routers directly work on the metal layers in the layout to optimize a weighted number of total wirelength and via count. 2D routers append traditional single-layer global routers with layer assignment techniques. In single layer global routing, wirelength is optimized. Global router calculates wire connection for every net in a

manner to minimize congestion and wirelength. In layer assignment stage, 2D global routing solution is extended to multiple layers with least number of vias possible. 2D global routing solution remains unchanged in a sense all the connections have unchanged x and y location, in a 3D dimension of x, y and z axes.

Such separation of objectives could greatly simplify the 3D global routing problem and could significantly reduce runtime. But it typically leads to inferior solution due to the blindness to via reduction in the first stage of 2D routing. To overcome the obvious shortcomings, 2D routers tries to create a model to measure certain metrics that could indirectly get translated into via count in the following stage. Usually the metric is the number of bends for 2D routing solutions and adding such metric to optimize could greatly reduce the number of vias. Although concurrently optimizing wirelength and via count is a more interesting problem formulation, it is rarely used due to the following two reasons. First, it is much harder than simple layer assignment problem to solve. Second, if one can successfully figure out a good technique to concurrently optimize congestion, wirelength and via count, it could be used as a standalone 3D routing solution as well, not just layer assignment techniques.

Hu and Sapatnekar [10] gave a detailed survey for global routing algorithms. Recently, the global routing algorithms have been improved significantly with the ISPD 2007 and ISPD 2008 global routing contests held successfully. In the ISPD 2008 invited paper "The Coming of Age of (Academic) Global Routing", Moffitt *et al.* [11] presented the recent progress in the global routing area.

Before the two ISPD routing contests [12] [13], most academic global routers adopted R&R strategy but proposed different techniques to improve solution quality or speed. Kastner *et al.* [6] proposed a pattern routing scheme by using L-shaped and Z-shaped patterns to speed up the routing. Hadsell and Madden [14] studied cost function used in global routing and proposed to guide the routing by amplifying the congestion map with various congestion cost functions.

In ISPD 2007 global routing contest, several routers (BoxRouter 2.0 [15], Archer [16], NTHU-Route [17; 18] and FGR [19]) employed a negotiation-based R&R approach which was introduced by PathFinder [20] and successfully applied to FPGA routing. The negotiation-based cost functions are used by maze routing to drive the net away from the consistently congested regions. In both ISPD 2007 and ISPD 2008 global routing contests, 3-dimensional benchmarks include the costs on vias for performance evaluation encourage the global routers to minimize wirelength together with via count.

Almost all recent global routers (BoxRouter 2.0 [15], Archer [16], MaizeRouter [21], NTHU-Route [17; 18] and default algorithm in FGR [19]) adopts the 2D global routing and layer assignment framework. Although the direct 3D techniques should produce better solution in theory, in practice it is less successful mainly due to their excessively long runtime. There are two academic 3D global routers, FGR [19] [22] and GRIP [5]. While FGR sequentially uses 3D maze routing to reroute nets for an existing routing solution, GRIP employs integer linear programming (ILP) to select 3D routing solutions. The majority of academic global routers belong to the second category. NTHU-R 2.0 [18], NTUgr [23], FastRoute 4.0 [24], NCTU-R [25] and BoxRoute 2.0 [15] all use a two-stage framework to solve the global routing problem. Best of the 2-stage global routers can generate solutions with quality on par with solutions generated by FGR, although still a little behind GRIP.

Beside GRIP, top performers of academic global routers all adopt the rip-up and reroute framework and numerous sequential routers proposed during or after the contests all tap into one single idea to enhance their congestion reduction capability. NTHU-R [17] [18], FGR [19], MaizeRouter [21], Archer [16], NTUgr [23] all use a negotiation-based cost function originally proposed for FPGA routing [20] to help the convergence. Even though FastRoute 3.0 [26] contrives a similar tuning free history-based Virtual Capacity Adjustment method, its performance is rather poor. Despite of the evolution of the cost function, the rip-up and reroute framework stays as a trial and error approach. The

potential to further exploit the framework is depleting.

There are also other approaches that work on routability issue. Westra et. al. [27] [28] proposed alternative framework that calculates congested region based on probability theory. The way they calculate routing probability ignore that fact that segments for a routed net should be connected, which is a major drawback. As a result, the probabilistic framework shows no advantage over sequential rip-up and rerouting approaches in solution quality. Besides, Chu et. al. proposed a look-up table based Steiner Tree generation technique and package called FLUTE [29] [30] that provides optimal Rectilinear Minimal Steiner Tree (RMST) for nets with up to 9 pins and good RMST solution for nets with more than 9 pins. FLUTE enables global routers to use very good topology to start with and greatly improves global routing solution. Effort needs to be taken not only in the routing stage but also in the earlier placement stage. Placement determines the instances and pin locations and poses great constraints on the routing solutions. Pan and Chu [31; 33] pointed out that a placer integrated with an efficient global router to model interconnects could help the placer to generate routing friendly placement results. To achieve this kind of integration, global router has to be fast enough to be invoked frequently to estimate the interconnections. Therefore, a high-quality global router also need to be fast to address the routing task requirements.

This thesis works on congestion elimination and wirelength/via optimization in global routing. It propose three major pieces of technology that improve global routing performance from three different aspects. The first work is FastRoute 4.0, an extremely efficient sequential global router that lies as the foundation for the third place winner for the ISPD 2008 global routing contests. The second contribution is a an efficient auction algorithm based pre-processing framework (APF) for global routing that pre-determines detour concurrently before any routing to ease the burden of sub-optimal and slow maze routing used in rip-up and reroute stage. Last but not least, this thesis proposes the very first hierarchical 3D global router that pushes sequential router to its limit so that it can

achieve the solution quality of full 3D global routing yet matching the quick runtime of 2D global router and the layer assignment framework.

The first work, FastRoute 4.0, integrates novel techniques to minimize via count in a 2D global routing plus layer assignment framework. Our key contributions are:

- A via-aware Steiner tree generation technique to form good starting topologies for multi-pin nets.
- A 3-bend routing technique to quickly explore the routing paths between a source pin and a sink pin with creating less vias.
- A 2 layer global routing grid graph model that estimate the number of vias with ease.
- A spiral layer assignment technique to map a 2D routing solution into its 3D counterpart.
- A flow to apply all above techniques to perform 3D global routing effectively and efficiently.

In order to eliminate congestion in a more consistent manner, the second work in this thesis propose an efficient pre-processing framework for global routing to simultaneously detour for 2-pin nets that interact with highly congested locations. The detour technique creates a detour edge for a 2-pin net and mandates that routing for the 2-pin net has to use the detour edge. In this way, APF can effectively detour a net to reduce congestion while preserving the flexibility of the routing solution. The pre-processing framework draws lesson from the market mechanism in economy theory. One of the fundamental principles of economy theory recognizes that a fully competitive market will optimally allocate resources and maximize social welfare. Our work designs an efficient market to let routing nets compete for precious routing resources to achieve better global routing solutions. The key contributions of the second work include:

- A detour pre-processing framework that greatly eases the rip-up and reroute effort to remove congestion. The new framework provides a new way to handle congestion elimination problem in global routing.

- An interval overflow lower bound technique that accurately computes the most congested interval for global routing. It provides a more realistic hot spot detection method for global routing.

- An auction based detour algorithm that efficiently determines detouring nets and locations, considering important factors like wirelength minimization, congestion reduction.

The flow of the detour pre-processing framework APF is shown in Fig. 1.1. Assume APF has a set of given topology and multi-pin nets are broken down into 2-pin nets, it first identifies the most congested interval. Then APF simultaneously chooses the nets to detour, calculate the detour locations nd create the detour pins for them using auction algorithm in step 2. For each detoured 2-pin net $A \sim B$ with a detour edge $p_1 \sim p_2$, APF will break the original 2-pin net into three parts: the detour edge and two new nets $A \sim p_1$ and $p_2 \sim B$. The procedure is repeated until there exists no significant congestion. The modified 2-pin net list is fed into global router. As a result, the new global routing flow can significantly improve the global routing wirelength and accelerate the convergence of global routing. The newly proposed pre-routing framework is compatible with any global router.

The above two works greatly improves routability and via count in global routing. But It is still natural to wonder whether anything can make 2D routers to take a leap to generate solutions as good as 3D routers or whether it is possible to dramatically cut down 3D routers' runtime to the level of 2D routers. The recent developments of 2D routers tend to disprove first attempt. Efforts to improve 2D maze routing by Liu et al. [25] or layer assignment by Lee et al. [34] only bring in marginal improvements. It seems that

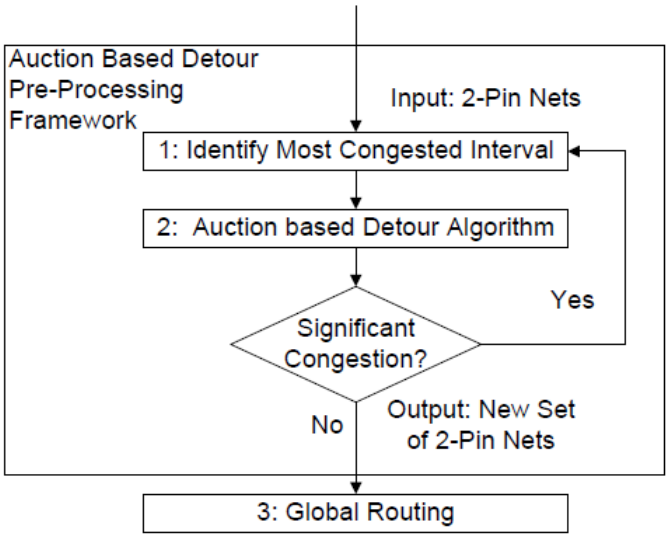


Figure 1.1 Global Routing Flowwith APF

directly constructing routing solutions on multiple metal layers is indispensable for high quality global routers. To get the combination of short runtime and short wirelength, I propose a multi-level 3D global routing framework MGR to achieve high solution quality and keep runtime in check. MGR exploits 3D maze routing to explore a much larger solution space but formulates it on the coarsened grid graph to get small problem size.

Multi-level framework is a quite mature concept in physical design, commonly used to speed up partitioning and placement tools. Multi-level framework has also been used in gridless routing by Cong et. al. in MARS [35] and Chen et. al. [36]. Gridless routing includes the task of both global routing and detailed routing. It is now rarely used since the routing problem becomes too complex to handle all at once. Detailed routing nowadays is typically treated as a stand-alone problem to handle myriads of design rules.

This is the first academic multi-level global router and name it MGR. MGR uses pattern routing and layer assignment to initiate a 3D solution. Then it uses a coarsening-uncoarsening multi-level framework to reroute nets in the congested region. Rerouting on coarsened grid graph greatly speeds up MGR while 3D routing provides better solution quality by exploiting the entire 3D solution space. The two factors balance out and lead

MGR to improve the performance and runtime of global routers. The key contributions of MGR include:

- A multi-level 3D global rerouting framework to efficiently generate high quality solutions. Unlike previous multi-level gridless routers, the framework starts with initial 3D solution which is refined by only one round of coarsening and uncoarsening process.
- An adaptive resource reservation technique in coarsening process, which provides accurate routing resource calculation for rerouting on coarsened grid structure to guide routing in higher levels.
- A new 3-terminal maze routing algorithm to optimally connect 3 separate terminals for a net. It provides better routing solutions for multi-pin nets comparing to traditional 2-terminal maze routing based technique.
- A network flow based solution propagation technique in uncoarsening stage, which introduces certain degree of concurrency to achieve better solution refinement in wirelength, congestion and the number of vias together, comparing to pure sequential method.

The rest of this paper is organized as follows. Section 2 introduces the global routing grid graph model. Section 3 details the key techniques used in FastRoute 4.0. Section 3 gives a full description of how the underlying principle and techniques used in the pre-processing detour global framework for global routing. Section 4 presents the algorithms proposed and integrated to form an unique 3D hierarchical global router. The experimental results are provided in Section 6 and this thesis concludes with Section 7.

CHAPTER 2. GLOBAL ROUTING GRID MODEL

Since global routing usually has a very tight schedule to finish and does not need to generate the detailed wiring geometries, it uses a simplified grid graph to represent the entire layout and abstracts away all design rules.

During global routing, complex design rules are abstracted away and a design is captured in a grid graph. As illustrated in Fig. 2.1, each layer of the entire routing region is partitioned into rectangular regions called global cells, each of which is represented by one node in the grid graph. The boundary on each metal layer between two global cells is represented by one 3D grid edge in the grid graph on the specific layer. The capacity for a grid edge, i.e., c_e, is defined as the maximum number of wires that can cross the grid edge. The usage, i.e., u_e, is defined as the actual number of wires crossing the grid edge. The overflow o_e is defined as $\max(u_e - c_e, 0)$. In the 3D model, a via is defined as a segment of wire that vertically connects one metal layer to a neighboring layer. 3D capacity and usage represents the routing resources and usage on each metal layer. The partitioning of metal layers into 3D global cells are synchronized so that cells on the same planar positions are aligned in the up-down directions. Capacities and usage for 2D gird graph are similarly defined as their 3D correspondents.

Wiring on the grid graph is greatly simplified. Only cross global cell connections are considered in the global grid graph model. Nets with all the pins residing in a single global cell are ignored in global routing process while multiple pins in the same global cells belonging to a single net are replaced with a single pin inside the specific global cell. All design rules are abstracted into capacities and usage on the grid graph. Every net is

considered to have its pins located at the center of global cells. Thus,the wirelength is discretized into the number of global cells boundaries each net will cross in routing.

The goal of global routing is to connect all cross global cell nets with no congestion, minimal wirelength and via. Eliminating congestion comes as the first priority because congested global routing solution may fail the detail routing process and the resulting chip could not be manufactured. Wirelength and via, though important, does not determine whether a design could be manufactured or not. But still, wirelength and via has big impacts on timing results and via is a determining factor in circuit yields. Their number should be minimized to enhance circuit performance.

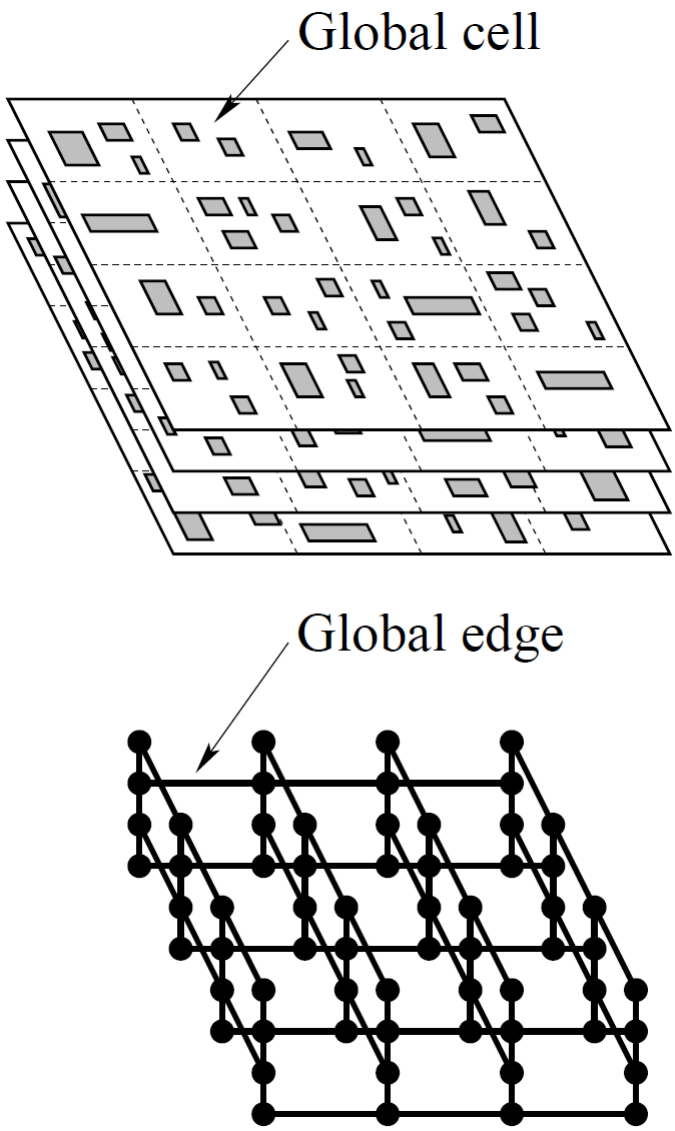


Figure 2.1 Global cells and corresponding 3D global routing grid graph.

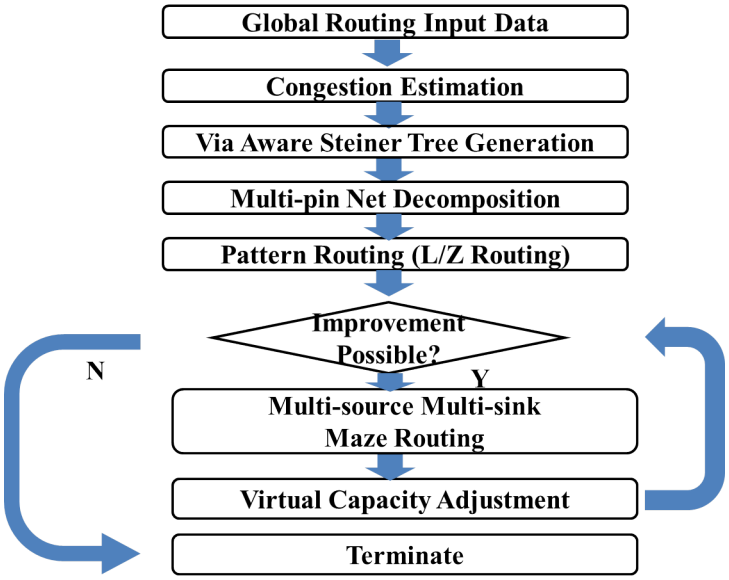
CHAPTER 3. FASTROUTE 4.0: A HIGHLY EFFICIENT 3D GLOBAL ROUTER

This chapter presents the detail insights and techniques used in FastRoute 4.0. FastRoute 4.0 was based on the FastRoute framework proposed by Min Pan, Chris Chu and Yanheng Zhang in FastRoute, FastRoute 2.0 and FastRoute 3.0. The framework provides a flexible, robust and efficient global router to start with. They focus on congestion elimination and improving efficiency for global routers, as the name FastRoute suggests. Yet, none of the three previous works in FastRoute consider how to reduce via count. This leaves room for FastRoute 4.0 to explore, especially in the area of via minimization.

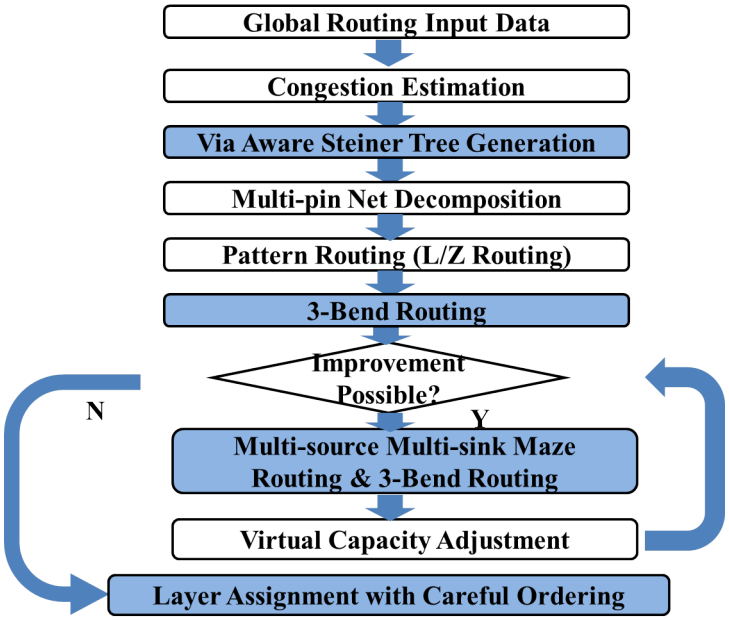
FastRoute 4.0 proposes three novel techniques to reduce via count mainly from three aspects:

- Via-aware Steiner tree generation technique to adjust net topology to minimize via count.
- 3-bend routing to replace monotonic routing and partially maze routing to reduce via count and improve global router's speed.
- Dynamic programming based layer assignment technique that carefully consider net routing structure and routing resources to minimize via count.

FastRoute 4.0 integrates these techniques into existing FastRoute framework and achieves significant improvements in both via count and runtime without much sacrifice in runtime and congestion elimination. It won the third place in ISPD 08 global routing



(a) FastRoute 3.0 flow



(b) FastRoute 4.0 flow

Figure 3.1 Comparison of routing flow between Fastroute 3.0 and FastRoute 4.0

contest. Fig. 3.1 gives the flow comparison between FastRoute 3.0 and FastRoute 4.0. It illustrates a flow that considers via throughout the entire global routing process, which is exactly what FastRoute 4.0 wants to achieve.

3.1 Via Aware Steiner Tree

As most global routers start to consider vias only in the late stages as maze routing and layer assignment, they overly rely on these techniques to solve congestion and reduce via count, where via reduction is often compromised. Moreover, in a common ripup and reroute framework, global routers avoid rerouting nets as much as possible to save runtime. So the majority of nets do not go through ripup and reroute process. These nets keep the original topologies, which are not optimized in terms of via count. Thus, an early consideration of vias in tree topology generation is essential for the quality of global router. Our via aware Steiner tree generation technique further extends congestion driven Rectilinear Steiner Minimal Spanning tree proposed in FastRoute[31]. It computes suitable topology at the beginning stage of global routing so that global router will generate less number of via.

After analyzing net topologies, I found that different tree topologies have significant impact on via count. As shown in Fig. 3.2, three topologies are generated for a 7-pin net. Assuming horizontal wires are on metal layer 1 and vertical wires on metal layer 2, the three topologies will generate 7, 14 and 9 vias respectively, ignoring the contacts between poly-silicon and metal layers. Here two special structures are defined: Horizontal Tree (H Tree) and Vertical Tree (V Tree). H tree is defined as a rectilinear tree with only one vertical trunks, with all the other trunks in between the vertical trunks and pin nodes to be horizontal. Similarly, vertical tree is defined as a tree with only one horizontal trunk, with all the other trunks in between the horizontal trunk and pin nodes to be vertical. If each net is assigned onto two adjacent metal layers, as our layer assignment

algorithm tries to achieve by keeping segments in one net close to each other, H Tree and V Tree are two extremes in terms of the number of vias. Other trees, like the RSMT with smaller wirelength shown in Fig. 3, have via count in between. However, it is not always the case that H Tree would have less number of vias than V Tree. If the resource on metal layer 1 is used up and the net has to go onto layer 2 and 3, it is obvious that V Tree is a better choice.

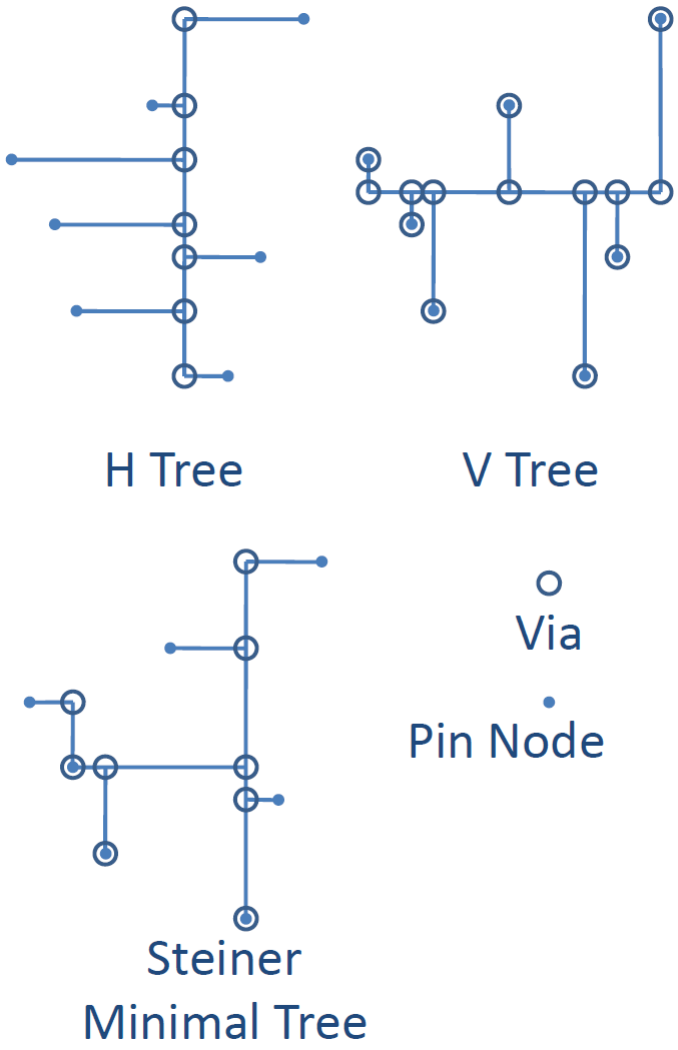


Figure 3.2 Via Aware Steiner Tree

So FastRoute 4.0 adjusts the net topology by the usage and capacity comparison between horizontal metal layers and vertical ones, as $\frac{\sum cap_{(h)}}{\sum usg_{(h)}} / \frac{\sum cap_{(v)}}{\sum usg_{(v)}}$, where $\sum cap_{(h)}$ and

$\sum usg_{(h)}$ is the sum of horizontal capacity and usage of grid edges within the bounding box of each net. $\sum cap_{(v)}$ and $\sum usg_{(v)}$ are defined for vertical edges. FastRoute 4.0 multiplies this factor with the factor used in FastRoute[31] that generates congested driven RSMT and use the result to scale vertical distances between the pin nodes. Finally, FastRoute 4.0 uses the scaled distance in FLUTE [29] [30] to generate adjusted topology for each tree. Trees generated in this way will take consideration of both via count and congestion. For example, if the horizontal resources are more abundant than vertical resources, it scale down the vertical distances. The RSMT computed by FLUTE for such an scaled net will have more horizontal edges. In this way, FastRoute 4.0 manipulate the constitution of horizontal edges and vertical edges in the net structure to reduce via count. The simulation shows a 3% via count reduction after pattern routing stage with less than 1% overhead in wirelength and no overflow overhead.

3.2 3-Bend routing

The most commonly used routing techniques in global routing includes L/Z/U pattern routing, monotonic routing and maze routing, as shown in Fig. 3.3. L/Z/U pattern routing generates limited number of via, has fast speed but is very limited in reducing congestion. Monotonic routing and maze routing, on the contrary, do better job in solving congestion problem but cannot control via count effectively. Besides, maze routing and U routing allow detour to strengthen the congestion reduction capability. Maze routing is most powerful but suffers from long runtime. So the traditional routing techniques all sacrifice one or several quality to improve some others. To address this problem, FastRoute 4.0 proposes 3-bend routing, a fast routing technique with enhanced congestion reduction capability than traditional pattern routing and less via than maze routing.

A 3-bend route is a 2-pin rectilinear connection that has at most three bends and

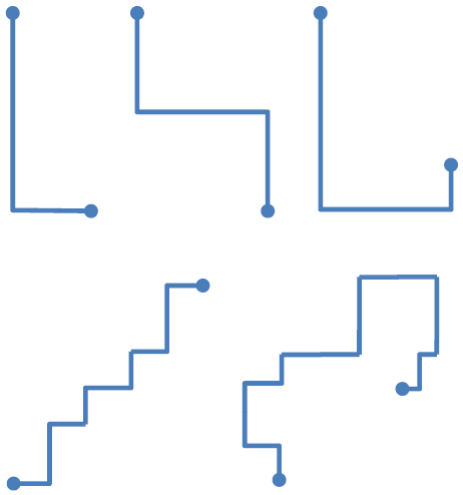


Figure 3.3 L/Z/U, monotonic and maze routing.

possible detour. It is much more flexible than L/Z/U route on solving congestion problem. Comparing to monotonic route [32] and maze route, 3-bend route has advantage on having less vias. Fig. 3.4 shows two possible 3-bend routes for a tree edge, $S \rightarrow B \rightarrow T$ and $S \rightarrow B' \rightarrow T$. No L/Z/U routing can avoid the congested area marked as shades. However, the 3-bend route $S \rightarrow B \rightarrow T$ can achieve congestion free routing with least bends possible.

To find the best 3-bend routing path for a 2-pin net, let us assume one pin to be the source ($S = (x_s, y_s)$) and the other one to be the sink ($T = (x_t, y_t)$). Without loss of generality, it could be assumed that S is at the lower-left corner and T is at the upper-right corner. The possible detouring region is defined as an expanding box for each net. It is calculated depending on the size, location and congestion of each net. The larger net with more congestion will have a larger expanding box. The pseudo code to compute the best 3-bend path for an S-T bounding box of size $p \times q$ and an expanding box of $m \times n$ nodes is given in Fig. 3.5.

In the algorithm, $d_h(x, y)$ and $d_v(x, y)$ denote the costs for a path going from the point (x, y) horizontally to the left boundary and vertically to the bottom boundary respectively. To balance wirelength and congestion, FastRoute 4.0 uses the same cost

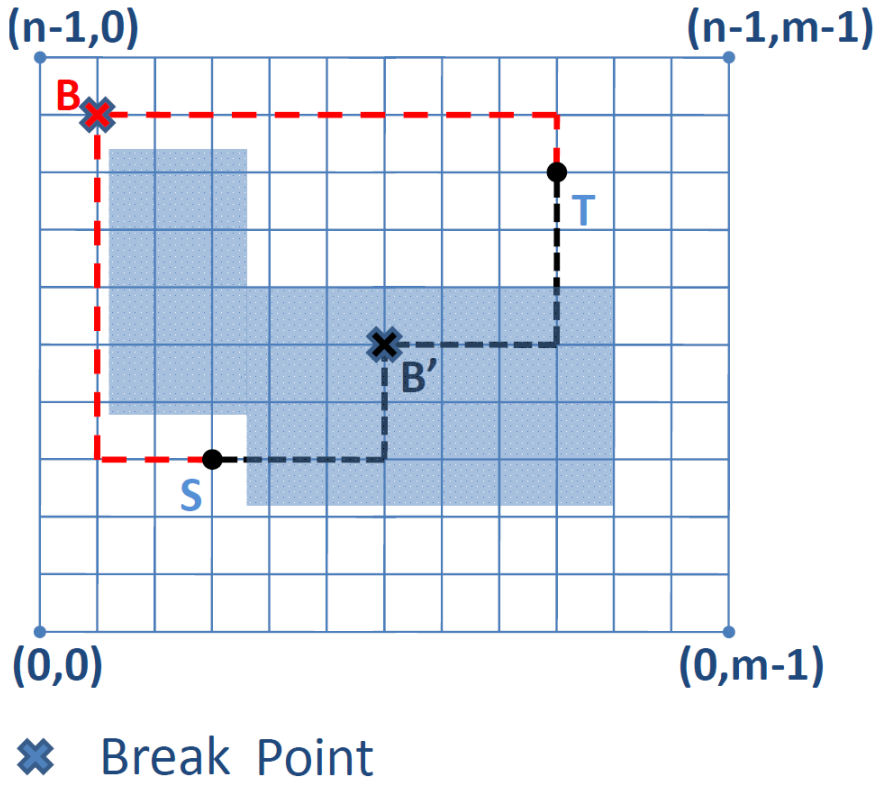


Figure 3.4 3-Bend routing.

Algorithm 3-Bend Routing

1. $C_{best} = +\infty$
2. **for** $y = 0$ to $n - 1$
3. $\quad d_h(0, y) = 0$
4. $\quad$ **for** $x = 1$ to $m - 1$
5. $\quad\quad d_h(x, y) = d_h(x, y - 1) + cost_h(x, y - 1)$
6. **for** $x = 0$ to $m - 1$
7. $\quad d_h(x, 0) = 0$
8. $\quad$ **for** $y = 1$ to $n - 1$
9. $\quad\quad d_v(x, y) = d_v(x - 1, y) + cost_v(x - 1, y)$
10. **for** $y = 0$ to $n - 1$
11. $\quad$ **for** $x = 0$ to $m - 1$
12. $\quad\quad B = (x, y)$
13. $\quad\quad d_{L1}(B) = |d_h(S) - d_h(x, y_s)| + |d_v(x, y_s) - d_v(B)|$
14. $\quad\quad d_{L2}(B) = |d_h(S) - d_h(x_s, y)| + |d_v(x_s, y) - d_v(B)|$
15. $\quad\quad d_{L3}(B) = |d_h(T) - d_h(x, y_t)| + |d_v(x, y_t) - d_v(B)|$
16. $\quad\quad d_{L4}(B) = |d_h(T) - d_h(x_t, y)| + |d_v(x_t, y) - d_v(B)|$
17. $\quad\quad$ Compute the cost of four possible 3-bend paths (i.e., L1-L3, L1-L4, L2-L3, L2-L4) from the four L-paths above plus via cost and compare them to C_{best}. If better, update the best 3-bend path.

Figure 3.5 3-bend routing algorithm

function for 3-bend routing as in maze routing. Line 2 to Line 9 create two tables that have the cost for a bend-free edge between any points in the expanding box and the left or bottom boundary, from which the cost of a 3-bend path between any two nodes in the expanding box could be easily calculated. A 3-bend path could be concatenated from two L-shaped paths, like using $S \rightarrow B$ and $B \rightarrow T$ to form $S \rightarrow B \rightarrow T$. So FastRoute 4.0 adds a break point in the expanding box, calculate the cost of the induced L-shaped paths in Line 13 to 16, from which FastRoute 4.0 can compute the cost of all the possible 3-bend paths and find the best solution. Line 2 to 9 take $O(mn)$ time. Line 10 to 19 also take $O(mn)$ time. So the complexity of 3-bend routing algorithm for a 2-pin net is $O(mn)$, the same as Z routing. It is worth noticing that the algorithm shown in Fig. 3.5 may compute some paths with overlapping segments. But they will be automatically excluded because of their high cost.

The short runtime and good congestion solving capability let 3-bend routing to become an alternative for maze routing. In the past, only a small percentage of nets would be routed by maze routing but the statement fails to hold as the benchmarks become more complex. FastRoute 4.0 applies 3-bend routing for congested nets before maze routing and mixes it up with maze routing during the rip-up and reroute loop, which leads to runtime and via count reduction.

3.3 Spiral Layer Assignment with Careful Ordering

There are generally two methods to generate solution for 3D global routing benchmarks. One is running routing techniques and layer assignment concurrently. It overly complicates the problem and is rarely used. The other more popular way first projects the 3D benchmarks from aerial view, finds a solution for the 2D problem and expands the solution to multiple layers. This expansion is called layer assignment, which has significant impact on the number of vias for the final solution. To keep our global router

fast, FastRoute 4.0 proposes a sequential layer assignment algorithm that would assign the 2D solution into routing layers, from lower layers to higher ones. The layer assignment algorithm will not change the aerial view of 2D solution and thus keep the total wirelength. Besides, our algorithm keeps total number of overflow unchanged. Thus, if FastRoute 4.0 can find a congestion-free solution for the 2D global routing problem, it can find a valid solution for the original 3D problem.

In the algorithm, FastRoute 4.0 fist orders the net considering its total wirelength and number of pin nodes. Then it orders the edges in each net according to their locations in the net. Finally, it assigns layers using dynamic programming, edge by edge, net by net.

Due to the competition of different nets in the assigning sequence and greedy nature of layer assignment, careless early assignment causes later nets "hopping" among the layers and thus generates a large number of unnecessary vias. Smaller nets connecting nearby global cells are considered relatively local and should use lower metal layers. On the contrary, longer nets assigned to upper layers will encounter less hopping between layers and will use wider tracks on top layers to achieve better timing. Furthermore, I observed that nets with higher number of pins tend to cause more vias. So FastRoute 4.0 orders nets by increasing order of $\sum wl/\#Pins$, where $\sum wl$ is the total wirelength for a net. Thus, it keep nets with smaller total wirelength and higher pin count on the lower layers.

For each net, FastRoute 4.0 orders edges for the following reason. The only layer information for a net is that the pin nodes must go up to at least metal layer 1 to have metal connections. So it orders the edges in each net in increasing order of their distance to the pin nodes. Here, the distance is defined as the number of edges the two nodes in an edge have to traverse to reach the nearest pin node. it first assigns layers to the edges with 0 distance i.e., edges that have at least one pin node and move onto edges with larger distance. By such an order, FastRoute 4.0 is sure that at least one end of

each edge has the information that which layers the pin node ranges between. Thus, it starts assigning edges on the periphery of a net and continue inwardly.

As shown in Fig. 3.6, FastRoute 4.0 creates a "via grid graph" to assign each edge to metal layers. Each node on the graph is named as a "via node". Vertical edges represent the possible places to add via while the horizontal edges are constructed from the actual 2D path. FastRoute 4.0 pulls straight the original zigzagged 2-pin net to form the horizontal edges in the via grid graph and copy the capacity and usage of corresponding global edge from the edge grid graph. FastRoute 4.0 breaks the tree edge into global grid edges and assign them to layers one by one. Such breakdown enables us to keep the total number of wirelength and overflow of the 2D solution unchanged. Without loss of generality, it assume sources S_i on the very left column and targets T_j on the right. If it does not know the layer information about the ending node, layer 1 to L are all considered to be targets. Here, L is the layer numbers in a benchmark. Otherwise, the target is set to be the spanning range of the ending node.

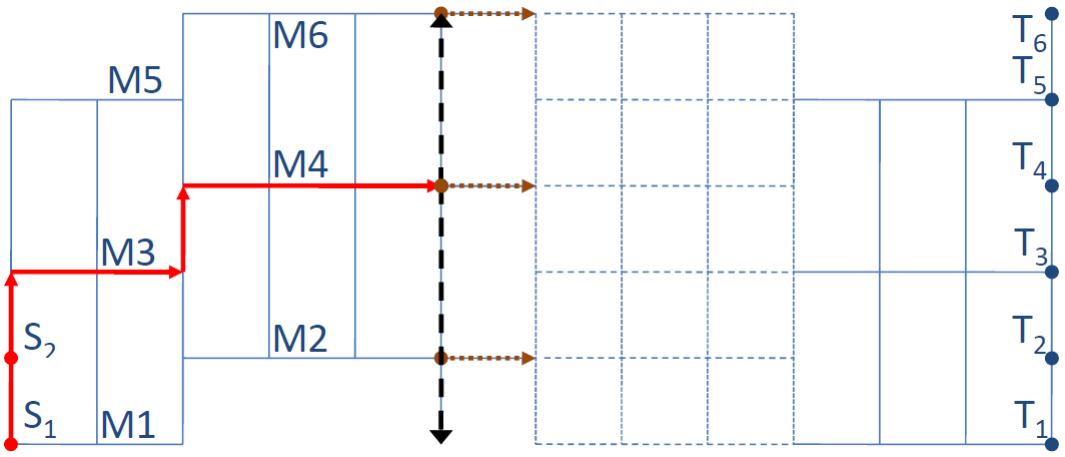


Figure 3.6 Dynamic programming layer assignment.

FastRoute 4.0 associates every via node with a cost, which represents the least number of vias on the paths from the node to any source nodes. Since it does not change the aerial view of a net, a 3D path must and must only use the horizontal segments between two adjacent columns once. Thus, the cost for a node is the same as its left neighbor if there

is still routing resource or one plus the cost associated with the upper or lower neighbor nodes, whichever is smaller. The pseudo-code to process each edge with wirelength n is shown in Fig. 3.7.

Layer Assignment for 2-Pin Net

1. Initial the cost for all the via nodes to $+\infty$
2. For every source, $C(j, 0) = 0$
3. Update the cost for other via nodes on the first column
4. **for** $x = 1$ to $n - 1$
5. **for** $j = 1$ to L
6. **if** $cap(j, x - 1) > usg(j, x - 1)$
7. $C(j, x) = C(j, x - 1)$
8. Update the cost from vertical neighbors.
9. Find the least cost for any sink node and trace back using $C(j, x)$

Figure 3.7 Layer assignment algorithm for 2-pin net.

In the algorithm, line 1 uses $O(nL)$ time and line 2 takes $O(L)$ time. The update of costs from vertical neighbors involves with a series of sorting, comparison and update, which takes at least $O(LlgL)$ time. However, because of the small number of L (typically less than 10 depending on the semiconductor process), FastRoute 4.0 could afford an $O(L^2)$ implementation. Hence, Line 4 to Line 8 take $O(nL^2)$. So the complexity of layer assignment for each edge is $O(nL^2)$.

3.4 Experimental Results

We implemented FastRoute in C with Steiner tree package FLUTE and the current version is FastRoute 4.1. All the experiments are performed on a Linux machine with 2.8 GHz Intel processor and 32GB RAM. We run experiments on ISPD08 global routing contest benchmarks [13]. The benchmark statistics are shown in table 3.1. It is worth

mentioning that FastRoute 4.1 now adopts a single set of tuning and avoids specific benchmark tuning to demonstrate the effectiveness of global routing framework and techniques presented in this work. On the contrary, all the participants in ISPD 08 contest use benchmark specific tuning.

The 2008 set of benchmarks has 8 new benchmarks and 8 benchmarks inherited from 2007. However, when ISPD08 global routing contest considers one unit of via at the same cost of one unit of wirelength, the one held in 2007 charges via at a cost three times of the cost for wirelength. In our experiment, we use the rules set by the 2008 contests which treats wire segments and vias equally.

Table 3.1 Experimental benchmarks statistics

Name	Grids	#Layers	#Nets	#Routed Nets	Max Deg	Avg Deg
adaptec1	324×324	6	219K	177k	340	4.2
adaptec2	424×424	6	260K	208k	153	3.9
adaptec3	774×779	6	466K	368k	82	4.0
adaptec4	774×779	6	515K	401k	171	3.7
adaptec5	465×468	6	867K	548k	121	4.1
newblue1	399×399	6	332K	271k	74	3.5
newblue2	557×463	6	463K	374k	116	3.6
newblue3	973×1256	6	552K	442k	141	3.2
newblue4	455×458	6	636K	531k	152	3.6
newblue5	637×640	6	1.26M	892k	258	4.1
newblue6	463×464	6	1.29M	835k	123	3.8
newblue7	488×490	8	2.64M	1.65M	113	3.6
bigblue1	227×227	6	283K	197k	74	4.1
bigblue2	468×471	6	577K	429k	260	3.5
bigblue3	555×557	8	1.12M	666k	91	3.4
bigblue4	403×405	8	2.23M	1.13M	129	3.7

Table 3.2 FastRoute 4.0 results on 3D version of ISPD08 global routing contest benchmarks

name	FastRoute 4.0					NTHU-R2.0 [18]			NTUgr [23]			BoxRouter2.0 [13]		
	ovfl	swl[1]	via[1]	twl[1]	cpu(s)	ovfl	twl[1]	cpu(s)	ovfl	twl[1]	cpu(s)	ovfl	twl[1]	cpu(s)
adaptec1	0	36.4	17.4	53.8	193	0	53.4	568	0	57.4	270	0	52.9	1227
adaptec2	0	33.3	18.9	52.2	51	0	52.3	98	0	53.7	66	0	52.7	162
adaptec3	0	96.7	34.5	131.2	183	0	131.0	510	0	135.0	264	0	131.8	1635
adaptec4	0	89.9	31.4	121.3	61	0	121.7	121	0	123.7	72	0	122.1	403
adaptec5	0	104.1	51.7	155.8	407	0	155.4	1077	0	159.9	918	0	156.9	1889
newblue1	0	24.5	21.8	46.3	361	0	46.5	290	6	49.3	58650	44	47.5	74488
newblue2	0	46.7	28.5	75.2	40	0	75.7	57	0	76.9	36	0	75.9	109
newblue3	31532	76.5	31.3	107.8	1353	31454	106.5	5728	31024	188.3	53040	38958	109.1	82615
newblue4	142	82.9	47.6	130.5	2140	138	130.5	4525	142	143.8	67086	200	129.5	78225
newblue5	0	148.8	82.1	230.9	565	0	231.6	908	0	244.9	1230	0	232.9	1700
newblue6	0	103.7	73.8	177.5	598	0	176.9	847	0	186.6	1278	0	179.8	1785
newblue7	54	186.1	166.8	352.9	16888	62	353.5	6734	310	372.2	86730	208	358.6	84743
bigblue1	0	37.9	18.7	56.6	257	0	56.0	641	0	60.0	918	0	56.9	1147
bigblue2	0	49.3	41.6	90.9	457	0	90.6	397	0	91.2	14898	0	90.4	2346
bigblue3	0	78.9	51.1	130.0	114	0	130.7	235	0	133.5	240	0	131.3	380
bigblue4	138	121.3	108.9	230.2	2144	162	231.0	6159	188	242.8	24786	472	231.6	52644
Comparison[2]	1	\	\	1	1	0.998	1.001	1.75	0.994	1.038	23.99	1.25	1.0007	26.55

In Table 3.2., this work compares the performance of FastRoute 4.0 on the ISPD08 global routing contest benchmarks with the top 4 routers besides FastRoute 3.0. FastRoute 4.0 is the fastest router. For the four benchmarks that no one can successfully finish routing without incurring any overflow, FastRoute achieves lowest overflow for two benchmarks. Due to the fact that other groups do not disclose the details about the metal wirelength part and via part of the total wirelength, it only compares the total wirelength. Since no newer data is available for BoxRouter2.0 after the ISPD08 contest, the results for BoxRouter2.0 from ISPD08 global routing is quoted from contest results. All runtime are scaled to 2.8GHz.

Comparing to NTHU-R2.0, the 2008 ISPD global routing contest winner, FastRoute achieves 0.01% and 74% improvement for total wirelength and runtime respectively on the 12 routable benchmarks. Comparing to the 2^{nd} place winner, NTUgr, FastRoute 4.1 can finish routing one more benchmark without overflow and can achieve 3.8% less wirelength in 15X faster speed for 11 benchmarks that the two routers both successfully finished.

Via accounts for 26% to 47% of the total wirelength of FastRoute solutions to the contest benchmarks. Although via has higher resistivity and larger process variation which makes it much more important than before, I still think that congestion reduction is the most important function for global router. Both of the two recent global routing contests held by ISPD gave highest priority to the overflow of solutions for evaluating the performance of global routers.

Even though most global router that participated in the 2008 ISPD global routing contests have greatly improved over their earlier version in the 2007 contest, I observed that some routers still face two challenges. One is how to handle the congestion left in the final stages. Even though FastRoute 4.1 and NTHU-R2.0 successfully finished routing for newblue1, they both failed newblue4, newblue7 and bigblue4, with a residue

[1]Segment wirelength, via and total wirelength are in unit of $10K$.
[2]Wirelength and runtime comparisons are based on overflow-free benchmarks.

overflow of just less than 150. The huge runtime spent by NTUgr and BoxRouter2.0 on newblue1 showed the inability to solve the few final overflow. Another challenge is the effectiveness for the global routers to balance between reducing the number of overflow and extending wirelength. The conflict incurs due to the fact that one of the most efficient method to reduce congestion is detour, i.e. extending wirelength, which could however induce congestion in other areas. One important way to effectively control the trade-off is through cost function used in maze routing. Although cost functions evolve from step function to logistic function and the variants of logistic functions, the fact that global routers that generates shorter wirelength or longer wirelength can only reduce congestion to a similar level demonstrates that there is considerable potential for the academic global routers to improve in this area.

Table 3.3 Improvement Decomposition for Techniques in FastRoute 4.0

name	FastRoute 4.1			No Tree Adj			No 3-Bend			Input Order LA		
	ovfl	twl	cpu(s)	ovfl	twl	cpu(s)	ovfl	twl	cpu(s)	ovfl	twl	cpu(s)
adaptec1	0	53.8	193	0	54.1	211	0	54.2	192	0	58.6	176
adaptec2	0	52.2	51	0	52.4	56	0	52.4	58	0	56.7	37
adaptec3	0	131.2	183	0	131.8	195	0	132.2	189	0	140.5	155
adaptec4	0	121.3	61	0	121.5	60	0	121.3	62	0	129.4	30
adaptec5	0	155.8	407	0	156.5	448	0	157.1	457	0	171.6	315
newblue1	0	46.3	361	0	46.4	335	0	46.4	313	0	52.2	299
newblue2	0	75.2	40	0	75.4	46	0	75.1	40	0	83.0	15
newblue3	31532	107.8	1353	33628	107.8	3895	38563	107.5	4345	31532	114.3	3737
newblue4	142	130.5	2140	146	130.9	2211	144	130.9	2644	142	139.9	2189
newblue5	0	230.9	565	0	231.9	646	0	232.2	670	0	254.8	446
newblue6	0	177.5	598	0	179.0	703	0	179.2	668	0	202.1	514
newblue7	54	352.9	16888	80	354.8	17376	86	353.3	22284	54	403.3	12633
bigblue1	0	56.6	257	0	57.2	300	0	57.0	383	0	63.6	231
bigblue2	0	90.8	457	0	91.1	516	0	91.1	762	0	99.1	434
bigblue3	0	130.1	114	0	130.2	115	0	130.4	242	0	150.0	73
bigblue4	138	230.2	2144	144	232.1	4059	142	231.5	5529	138	265.5	2807
Comparison[3]	1	1	1	1.38	1.005	1.23	1.07	1.004	1.10	1	1.11	0.83

To demonstrate the effectiveness of the global routing techniques proposed in this paper, each major technique is turned off to see the performance degradation as shown in Table 3.3. In the column "No Tree Adj", it turns off the congestion-driven via-aware Steiner tree generation and use unadjusted tree topology directly generated from FLUTE. This configuration of FastRoute leads to 38% more congestion and 23% run time overhead. The "No 3-Bend" column shows the performance of FastRoute without 3-Bend routing. The degradation shows up for all three qualities focused on, though the degradation are not very significant. However, FastRoute spends 55% more runtime for the four unroutable benchmarks without 3-Bend routing, which has explanation in the fact that 3-bend routing is much more efficient than maze routing. For the last configuration, net ordering and segment ordering used in the spiral layer assignment is turned off and it shows that the two ordering saves 11% of wirelength, which would translate into significantly more percentages of via.

[3]Wirelength and runtime comparisons are based on overflow-free benchmarks.

CHAPTER 4. APF: PRE-PROCESSING FRAMEWORK FOR GLOBAL ROUTING: AUCTION BASED DETOUR TECHNIQUE

With FastRoute 4.0 at hand to minimize via count in global routing, congestion elimination again becomes the bottle neck for global routers. I made the statement because of the two following reasons. Firstly, during the development of FastRoute 4.0, global routers in general are quite sensitive to tuning parameters. Different test cases used in ISPD08 global routing contests usually requires quite different tuning profile to work properly. Such heavy reliance on tuning parameters show that ripup and rerouting process need to work in quite different behavior to eliminate different kinds of congestions. Since design is keep on changing, tuning could be very burdensome, if it works at all for difficult cases. Secondly, FastRoute 4.0, together with other academic global routers, spends a substantial part of runtime to eliminate the final few overflow. It is typical for a router to reduce overflow in initial routing solution from level of tens of thousands to down to a few thousand in 10 minutes but spends an hour to totally eliminate the final couple of thousands overflow. Both of the phenomenon shows that FastRoute 4.0 and other academic sequential routers over rely on maze routing's greedy behavior. Although it is optimal for routing a single net, it lacks a global view and sets great limit to its performance. Global routing needs a more systematic method to eliminate congestion.

Thus the second work in this thesis is a auction algorithm based pre-processing framework (APF) to provide guidance for following global routing stage to eliminate conges-

tion. It first detects most congested region in layout and then use game theory to let nets compete for resources in the congested region. Based on the results, it changes the topology for the input net to predetermine detour and send such detour information for global router to process.

4.1 Interval Overflow Lower Bound

In the detour pre-processing framework, APF needs to first identify the most congested locations.

One simple option for us is the currently available congestion estimation techniques. This would lead us to the dominating probabilistic congestion estimation (PCE) techniques. PCE is currently used to charge higher cost for supposedly congested region [14] or to bias tree topology with less congestion [31]. Nevertheless, using PCE is inaccurate for the purpose. One of the major reason is the different behavior between a global router and probabilistic congestion estimator. Global routers tend to use less congested region so there is preference for certain routing choices. On the contrary, PCE treats every possible routing equally [37] and this causes biased estimation.

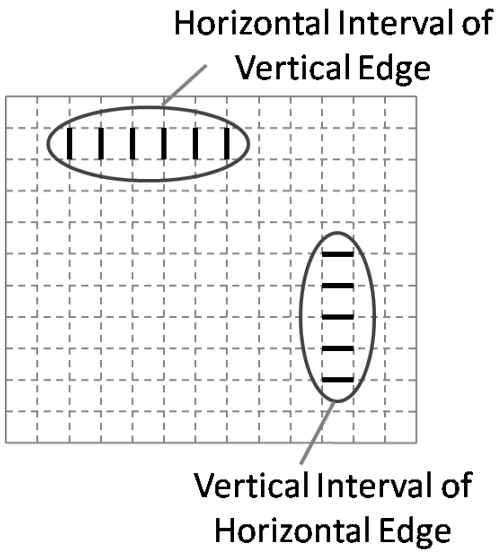


Figure 4.1 Intervals of Grid Edge

Instead of PCE, APF propose an interval overflow lower bound (IOLB) technique that calculates the lower bound of overflow on an interval. An interval is defined as a horizontal sequence of neighboring vertical grid edges or a vertical sequence of neighboring horizontal grid edges, as shown in Fig. 4.1.

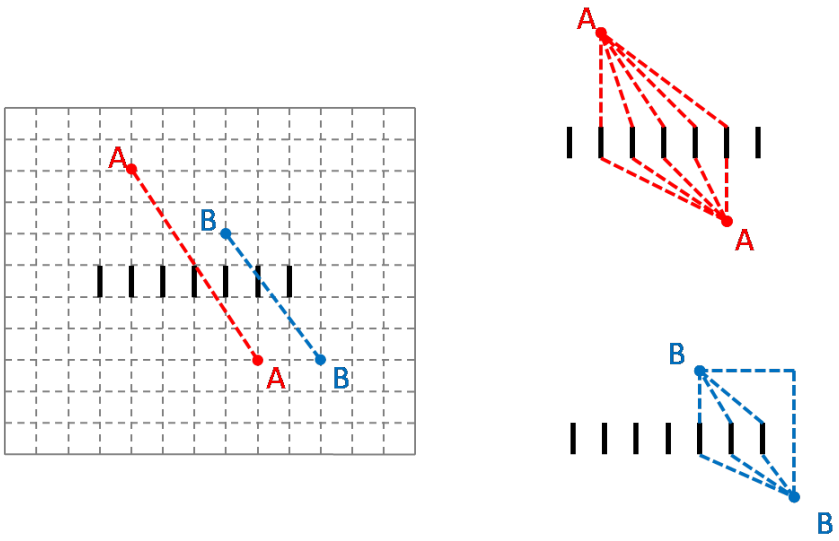


Figure 4.2 Net-Interval Intersection

A net is called fully intersecting an interval if the bounding box of the net crosses the interval twice. In Fig. 4.2, net A fully intersects the interval but net B does not. Without detour, net A has to use one grid edge in the interval while net B has other choices. APF defines the demand D_I for an interval I as the total number of 2-pin nets that fully intersects the interval. D_I is the lower bound demand under the conditions of fixed topology and no detour. These two conditions generally hold for sequential global router when it generates tree structure and uses pattern routing to initialize the routing solutions. If the lower bound demand exceeds the total capacity C_I of the interval, then it is for sure that some nets have to detour. So APF deducts the capacities of the interval from the lower bound demand to generate interval overflow lower bound O_I, where $O_I = D_I - C_I$. APF needs to detour at least O_I nets that fully intersect I so that there would no longer be any congestion on the grid edges in the interval.

For a 2-pin net $(y_1, x_1) \sim (y_2, x_2)$, without loss of generality, we assume $y_1 < y_2$, $x_1 < x_2$ and denote $q = x_1 - x_2$. The 2-pin net has impacts on every horizontal interval between row y_1 and $y_2 - 1$ that includes interval $x_1 \sim x_2$. If APF goes through every 2-pin net and updates its demand on every interval each net fully intersects, the run time would be $O(mn^3)$ where m is the number of nets and n is the maximum side size of global routing grids. This is too slow for large scale design.

Algorithm: IOLB for Horizontal Intervals

1. $\forall y, \forall x, \forall q, O_h[y][x][q] = 0$
2. **for** every 2-pin connection $(x_1, y_1) \sim (x_2, y_2)$
3. $\quad x_{min} = min(x_1, x_2)$, $x_{max} = max(x_1, x_2)$
3. $\quad y_{min} = min(y_1, y_2)$, $y_{max} = max(y_1, y_2)$
4. $\quad$ **for** $i = y_{min}; i < y_{max}; i++$
5. $\quad\quad O_h[i][x_{min}][x_{max} - x_{min}] += 1$
6. **for** $y = 0; y < yGrid - 1; y++$
7. $\quad$ **for** $x = 0; x < xGrid - 1; x++$
8. $\quad\quad aid_h[x][0] = O_h[y][x][0] - cap_h[y][x]$
9. $\quad\quad$ **for** $q = 1; q < x; q++$
10. $\quad\quad\quad aid_h[x][q] = aid_h[x][q-1] + O_h[y][x-q][q]$
11. $\quad\quad O_h[y][x][0] = O_h[y][x][0] - cap_h[y][x]$
12. $\quad$ **for** $q = 1; q < xGrid; q++$
13. $\quad\quad$ **for** $x = 1; x < q - 1; x++$
14. $\quad\quad\quad O_h[y][x][q] = O_h[y][x][q-1] + aid_h[x+q][q]$

Figure 4.3 Interval Overflow Lower Bound Algorithm for Horizontal Intervals

IOLB for an interval I_A consisting of $e_1, e_2, \ldots, e_q$ equals to the IOLB of the interval I_B consisting $e_1, e_2, \ldots, e_{q-1}$, plus the number of nets that fully intersect I_A but not I_B, minus the capacity of the grid edge e_q. Based on the observation, APF proposes an

algorithm based on dynamic programming that computes IOLB for all the intervals in $O(n^3)$ time. The algorithm to compute IOLB of a horizontal intervals is shown in Fig. 4.3.

In Fig. 4.3, $xGrid$ and $yGrid$ is the size of the global routing grid. $O_h[y][x][q]$ will hold all the horizontal IOLB APF wants to compute. y denotes the intervals on the y^{th} row of vertical grid edges, counting from the lower corner. x sets the left boundary index of the interval and q is the length of the interval. In line 2 to 5, for each net, APF adds its demand to the shortest intervals it fully intersects in row y_1 to row $y_2 - 1$. Line 8 to 10 computes the number of nets that fully intersect interval $I[y][x][q]$ but not interval $I[y][x][q-1]$, minus the capacity of the edge $e[y][x+q]$. Line 11 initializes the IOLB for intervals consisting of a single grid edge. APF uses line 13 to add up the two parts together and arrives at the IOLB for an interval one grid edge longer than the previous one.

The three level of for loops from line 6 to line 13 set the complexity of the IOLB algorithm to $O(n^3)$. Because the routing grids generally have $O(n^3)$ intervals and IOLB computes all of them, there exists no superior algorithm with less complexity. APF can obtain the interval overflow lower bound for vertical intervals similarly.

The interval I^* with largest total overflow is denoted as the most congested interval because a global router has to detour O_{I^*} nets in order to eliminate congestion on the interval. So the larger O_{I^*} is, the more onerous task global router has.

IOLB technique with $O(n^3)$ complexity is not a trivial operation for large scale designs. Since the detour step that follows will change the routing solutions for nets fully intersecting the most congested interval, APF needs to update IOLB for impacted regions. APF uses regional updating technique to calculate the overflow only for intervals impacted by the detoured nets, instead of computing IOLB from scratch again. Although the regional update technique also has a complexity of $O(n^3)$, the size of the update region is much smaller than the entire global routing grids.

IOLB, by its name, is the lower bound for congestion on an interval. Although it may under-predict the overflow significantly for intervals with few fully intersecting nets, it rarely under-predict for an interval I with large overflow. Every net that does not fully intersect I would avoid using grid edges in the interval. Every net that fully intersect I has to use one grid edge in I if detour is banned. Thus APF accurately counts the nets that have no choice other than the grid edges in I. The resulting IOLB for I is actually the exact overflow.

4.2 Detour Problem

The interval overflow lower bound calculated in Sec. 4.1 gives the least amount of nets that need to eliminate congestion on the interval. All the detouring nets need to cross an extended interval of the original congested interval and the detour problem targets the issue that which nets should be detoured and where should the detoured net intersect the extended interval. The selection of detouring nets and the crossing locations need to consider the impacts of congestion in affected region and the extra wirelength caused by detours.

For the most congested interval I_C with D_{I_C}, C_{I_C} and O_{I_C}, APF extends the interval to find the shortest interval that fully contains I_C and can accommodate all crossing demands on itself. The new interval is denoted as Extended Interval(EI). Fig. 4.4 shows an example of the extension process. For a given IC, APF will try to include one of the two grid edges neighboring the boundary of the congested interval. The grid edge that leads to an extended interval with smaller IOLB will be chosen. It is called EI evaluation. After one grid edge is included, if the extended interval has positive IOLB, APF will extend EI to include the grid edge it does not choose in previous EI evaluation. In this way, APF can achieve almost even extension on both sides of I_C, which balances the detour on both sides and leads to shorter wirelength. If the new EI is still congested,

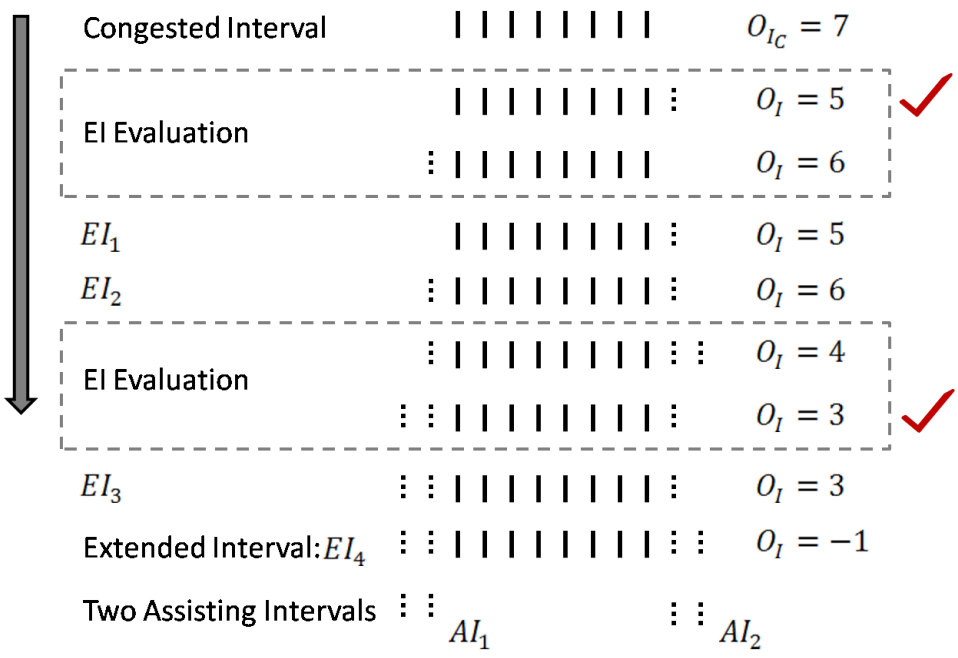


Figure 4.4 Extension for Congested Interval

APF will go back to EI evaluation process. The EI evaluation and the possible inclusion of the forsaken grid edge is repeated until APF finds an interval with non-positive IOLB. For the above example, EI_4 is the final Extended Interval APF wants to find. The extended portions of EI is called assisting intervals, denoted as AI_1 and AI_2.

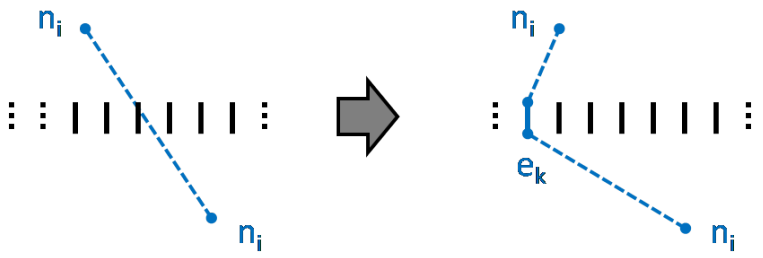


Figure 4.5 2-Pin Net Decomposition for a Determined Crossing

APF creates non-detour crossing sites on I_C and detour crossing sites on AI_1 and AI_2. Each crossing site corresponds to using one unit of routing usage. If one net n_i chooses one crossing site s_j on grid edge e_k, as shown in Fig. 4.5, its routing will consists of three components: the two smaller sub 2-pin nets and the usage of e_k. In the pre-processing

framework, APF doesf not calculate the routing of sub 2-pin nets since it does not want to overly constraint the routing solution. Because the pre-processing framework focus on detour that eliminates congestion, it will ignore the "non-detour" nets that use the crossing sites on I_C and only decompose "detour" nets to generate the new set of 2-pin nets.

Algorithm: Detour Site Creation

1. $min\_IOFL = O_{I_C}$
2. $i = 1$
3. **while** $min\_IOFL > 0$
4. $\quad ns_{e_i} = max(0, min\_IOFL - max(O_{EI_i}, 0))$
5. $\quad min\_IOFL = min(min\_IOFL, O_{EI_i})$
6. $\quad i++$

Figure 4.6 Detour Site Creation for Assisting Intervals

If the two assisting intervals consists of k grid edges, i.e. $e_1, e_2, \ldots, e_k$, indexed according to the sequence of their inclusion and the resulting extended interval after including grid edge e_i is labeled as EI_i, the number of detour sites created for e_i is calculated Fig. 4.6. The number of detour crossing sites created for e_i is basically $O_{EI_{i-1}} - O_{EI_i}$, the number of nets APF can detour on e_i without inducing congestion on e_i. If it uses $O_{EI_{i-1}}$ instead of min_IOFL to calculate the number of sites, for the example in 4.4, it would create 3 sites for e_3 and 8 sites in total. 8 is larger than the IOLB of I_C because using $O_{EI_{i-1}}$ will create one detour site to detour a net fully intersecting EI_2 for the example. But such detour should not be considered in the detour problem for I_C. Thus, it uses min_IOFL to guarantee that it will create O_{I_C} detour crossing sites. In line 4, there are two max operations. The first max is used to prevent creating any detour sites on an already congested e_i. The second max limits the detour sites to O_{I_C} because the final EI may have a negative IOBL and lead to excessive detour sites.

For a grid edge e_j in I_C, APF creates cap_{e_j} non-detour crossing sites. So in total, it create O_{I_C} detour crossing sites on assisting intervals and C_{I_C} non-detour crossing sites on I_C. For the detour problem, APF only considers the D_{I_C} nets that fully intersect the most congested interval since it wants to choose a subset of those nets form them to detour and determine their detour crossing locations with the assisting intervals. The number of crossing sites actually equals to the number of nets.

When crossing sites are created, each net and crossing site pair is assigned a weight which has the name "quality loss", denoted as ql and defined in Equation 4.1:

$$ql_{i,j} = \alpha \cdot detour + \beta \cdot congestion \tag{4.1}$$

In the above equation, *detour* is the extra wirelength caused by the detour of net n_i using site s_j. *congestion* models the impacts on congestion that would be caused by the two sub 2-pin nets after the net decomposition, as shown in Fig. 4.5. The congestion is calculated based on probabilistic Z routing presented in [37].

APF generates a bipartite graph in which every net has a node residing on one side of the partition and every crossing site is given a node residing on the other side. There exists an edge between every pair of nodes on the different sides and the edge weight is set as the "quality loss".

At first sight, it seems that APF can find a minimum weighted bipartite matching to achieve the optimal set of crossing locations for all the nets. The minimum weighted bipartite matching solution corresponds to the optimal crossings in terms of both wirelength and congestion impacts for the nets that cast demands on I_C . However, the detour problem can potentially involve a dense graph with up to a million edges so directly solving the bipartite matching would be very slow.

On a second thought, there is an analogy between economic market and global routing. Every net tries to use smallest possible wirelength to complete routing and nets using the same congested grid edge are competing to stay still to preserve the shortest

wirelength. If treating every net as a bidder, every crossing site as an object and allowing every net to seek after and compete for its desirable crossing sites, it would be like holding auctions to allocate routing resources of the extended interval. In fact, there are a few papers addressing the hidden link between auction algorithm and matching algorithm [38] [39]. So APF use auction algorithm to solve the detour problem.

4.3 Auction Algorithm

To reduce the problem size, APF only creates one crossing site for each grid edge on the extended interval, instead of multiple crossing sites on each grid edge. Each site can accept multiple bids and the number of acceptable bids is equal to the number of crossing sites it originally create for the grid edges. APF let the D_{I_c} nets fully intersecting I_C to compete for their desirable crossings over the extended interval.

The desirability for a net n_i to use the crossing sites on grid edge e_j is captured in the maximum price $w_{i,j}$ n_i is willing to pay to use n_i. APF uses "quality loss" defined in the last section to calculate $w_{i,j}$, the formula is shown below:

$$w_{i,j} = ql^* - ql_{i,j} + 1 \tag{4.2}$$

ql^* is the maximum ql. For every e_j, APF creates a priority queue of size ns_{e_j} to store temporary leading bidder. It denotes p_j as the price of e_j. The net benefit of an assignment Π is defined as

$$\sum_{n_i}(w_{i,\Pi(i)} - p_{\Pi(i)}) \tag{4.3}$$

The goal is to find the Π^* that maximizes the net benefit. The auction algorithm is an iterative method to find the optimal prices and an assignment that maximizes the net benefits.

The auction algorithm in [38] presents a two-step framework to achieve such optimal assignment. In the first step, every bidder calculates a price for one item it wants the most. In the second step, every item is assigned to its highest bidder. The two steps

are repeated until every bidder gets an item. However, the auction algorithm cannot be directly used here because every item can only accept one bid. So APF modifies the second step of the auction algorithm so that every item, the crossing site on every grid edge, can accept up to ns_e bids. The modified auction algorithm is shown in Fig. 4.7.

In the modified auction algorithm, ϵ is introduced to prevent endless cycles. If the value of ϵ is chosen properly, the assignment generated by the terminated auction algorithm will satisfy "ϵ complementary slackness", i.e. all nets are assigned to grid edges that are within ϵ of being best [39].

Generally, if an object receives a bid in m iterations, its pries is at least $m\epsilon$. For a sufficiently large m, the object becomes very expensive and every nets still not assigned will avoid it. The auction algorithm converges in $O(mw^*/\epsilon)$ iterations, where m is the number of auctioned grid edge and w^* is defined as $max_{i,j}|w_{i,j}|$. So the worst case complexity of the modified auction algorithm is thus $O(nm^2w^*/\epsilon)$, where n is the number of bidding nets. In practice, due to the variance of detour wirelength and congestion impacts, the increase in bidding price is much larger than ϵ. The auction algorithm runs quite efficiently.

For the original auction algorithm, in which every object accepts only one bid, [40] proved that the total net benefits is within $n\epsilon$ of being optimal. The modified auction algorithm can be proved to be within $n\epsilon$ of being optimal in a similar manner.

The modified auction algorithm is a relaxation solution for the original detour problem and it has several advantages. It generates a solution within the vicinity of optimal solution in theory. Our experimental results support such theoretical solution quality. On the other hand, the modified auction algorithm can be fully parallelized. The bidding price evaluation of each net is independent from other nets and the assignment procedure for every grid edge is independent from other grid edges. So the auction algorithm is very suitable for today's multi-core platforms. More importantly the auction algorithm provides us a method to simultaneously decide crossings and detour a minimum number

Algorithm: Modified Auction Algorithm

1. Initialize the assignment $\Pi = \emptyset$, the set of unassigned net $I = n_1, \ldots, n_n$, set prices $p_j = 0$ for all e_j.
2. The algorithm runs in two phases and repeats until I is an empty set
3. Phase I: Bidding for all $n_i \in I$
4. (1) Find benefit maximizing grid edge

   $j_i = argmax_j\{w_{i,j} - p_j\}$

   $j_i = max_j\{w_{i,j} - p_j\}$

   $u_i = max_{j \neq j_i}\{w_{i,j} - p_j\}$
5. (2) Compute the bid of bidder n_i to grid edge e_j

   $b_{n_i \to e_j} = w_{i,j_i} - u_i + \epsilon$
6. Phase II: Assignment for each grid edge e_j
7. (3) Let $B(j)$ be the set of new bidders from which e_j received a bid in the current iteration and $L(j)$ be the set of nets leading the bid for and currently assigned to it. If $|B(j)| + |L(j)| > ns_e$, only the ns_{e_j} highest bidder from $P(j) \cup L(j)$ will be assigned to e_j. If $|B(j)| + |L(j)| \leq n_{e\_s}$, all $n_i \in B(j)$ will be assigned to e_j. p_j will be set to the lowest bidding price of the accepted net and the priority queue for temporary leading bidders will be updated.

Figure 4.7 Modified Auction Algorithm

of nets to eliminate the congestion over the interval with largest IOLB.

4.3.1 Speed Up Techniques

The modified auction algorithm stated in the previous section has a complexity of $O(nm^2w^*/\epsilon)$. With a common auction problem involving an interval of up to a hundred grids and thousands of bidding nets, the computation can drastically slow down the entire global routing process. Because the pre-processing framework focus on detour, APF is much more interested in which nets fail to get non-detour sites and are forced to use detour sites. Thus, it uses the following simplification techniques to identify which nets to detour and where they should detour more efficiently.

The first type of simplification is congested interval abstraction. The nets with successful bid to use the original congested interval will not go through significant detour. Because the pre-processing framework focus on generating necessary detour to eliminate congestion, the non-detour crossings are purposefully ignored to maintain routing flexibility. The actual routing for non-detouring nets in the auction is left to global router to determine. Thus, it does not make much sense to compute where the non-detouring nets intersect with I_C. So APF combine the grid edges in the original congested interval into a single crossing site, with the ability to accept C_{I_C} bids. The quality loss for this object is computed by assuming no detour and minimum congestion impacts.

In addition, APF carries out a pre-selection procedure to limit the number of nets to participate in the modified auction. It is obvious that some nets will use a lot of detour or cross significantly congested region to use any detour crossing site on the assisting intervals. They can be ignored in the auction without sacrificing the solution quality. So APF selects $2O_{I_C}$ nets (if $2O_{I_C} < D_{I_C}$) with lowest "quality loss" to use the detour crossing sites. Meanwhile, the number of bids the non-detour site can accept is reduced accordingly to O_{I_C}.

The last type of speed up technique is side selection. The principle is very simple.

Some of the $2O_{I_C}$ nets will obviously use one of the assisting interval instead of the other due to detour or congestion. During the pre-selection process, APF can evaluate the quality losses derived from using the two assisting intervals. Comparing the two quality losses, APF categorizes the nets into three types: nets able to bid for assisting interval 1, nets able to bid for assisting interval 2 and nets able to bid for both assisting intervals. The side selection limits the net to bid for sites on one assisting interval and the non-detour site if they are not categorized to be able to bid for both intervals.

The three types of simplification together can reduce the problem size by more than 90% and allow us to use the auction based detour algorithm for a lot more congested intervals.

4.3.2 Solution Refinement

APF uses the assignment solution to modify the netlist of original testcases and feed the new set of 2-pin nets that contains detour decisions to the global router. One of our concerns is that the detour algorithm might hamper the flexibility of solutions. To avoid causing congestion in the neighboring region where detour sites are created, it designs a solution refinement stage to use maze routing to reroute the 2-pin nets that are split in the pre-processing stage. Such solution refinement brings in less than 0.1% wirelength improvement, which indicates that the original pre-processing framework is performing well. Thus, APF removed the solution refinement due to its inefficacy.

4.4 Experimental Results

The auction based pre-processing framework is implemented in C and use FastRoute 4.0 as the following global router. All the experiments are conducted on a Linux machine with a $2.6GHZ$ Intel Processor and $32GB$ memory. I compare the results of work with the winning teams in ISPD 08 global routing contests and the benchmarks used are from

ISPD 08 contest too.

The benchmarks are separated into two categories: routable benchmarks and unroutable benchmarks. The performance comparison for routable benchmarks is shown in Table 5.1. Comparing to the contest winners, our work achieves shortest wirelength using much less run time. Our work achieved 2.5%, 1.1% and 4.9% less wirelength comparing to FastRoute4.0, NTHU-R2.0 and NTUgr respectively.

The performance comparison for unroutable benchmarks is shown in Table 5.2. Again, our work uses least amount of time to generate solutions with shortest wirelength. It is worth noticing that our work also achieves the smallest number of remaining overflow for 3 benchmarks, which demonstrates the effectiveness of our simultaneous detour framework. For "newblue3", our work falls short by 104 nets or 0.3% from the best results generated by NTUgr. However, our work only spends 1.7% runtime of NTUgr for this specific benchmark.

Table 4.1 Comparison of Routing Results on Routable Benchmarks for APF

Name	Our Work		FastRoute 4.0 [24]		NTHU-R 2.0 [18]		NTUgr [23]	
	Wirelength	cpu(s)	Wirelength	cpu(s)	Wirelength	cpu(s)	Wirelength	cpu(s)
adaptec1	5352K	400	5460K	279	5344K	611	5740K	291
adaptec2	5151K	89	5277K	59	5229K	102	5370K	71
adaptec3	12922K	384	13213K	240	13101K	549	13500K	284
adaptec4	12021K	116	12249K	41	12169K	130	12370K	78
adaptec5	15394K	643	15866K	660	15538K	1160	15990K	988
bigblue1	5598K	333	5775K	530	5595K	690	6000K	1169
bigblue2	8919K	462	9352K	792	9059K	427	9120K	16044
bigblue3	12899K	203	13073K	158	13068K	253	13350K	258
newblue1	4575K	431	4686K	377	4653K	312	4930K	63161
newblue2	7459K	70	7636K	18	7570K	61	7690K	39
newblue5	22909K	595	23377K	777	23158K	977	24490K	1324
newblue6	17560K	565	18078K	884	17689K	912	18660K	1376
Comparison	1	1	1.025	1.12	1.011	1.44	1.049	19.8

Table 4.2 Comparison of Routing Results on Unroutable Benchmarks for APF

Name	Our Work			FastRoute 4.0 [24]			NTHU-R 2.0 [18]			NTUgr [23]		
	Overflow	WL	cpu(s)	Overflow	WL	cpu(s)	Overflow	WL	cpu(s)	Overflow	WL	cpu(s)
bigblue4	132	22887K	1431	150	25147K	3640	162	23090K	6633	188	24280K	26692
newblue3	31128	10615K	1749	31634	10752K	1181	31454	10653K	6168	31024	18830K	57120
newblue4	130	12964K	1208	140	13821K	2382	138	13046K	4873	142	14380K	72246
newblue7	54	35177K	7312	58	35974K	10209	62	35522K	7252	310	37220K	93401
Comparison	1	1	1	1.017	1.050	1.49	1.012	1.008	2.13	1.007	1.16	21.3

Table 4.3 shows the run time decomposition of our framework. For easy benchmarks, it spends a significant portion of runtime to calculate IOBL and host auctions while saving relatively little runtime from global router. As benchmarks become larger and harder, the advantage starts showing up. For hard benchmarks, IOBL and auction algorithms spend less than 20% of total runtime. Because the framework actually outruns FastRoute 4.0 by another 20% 40% for hard benchmarks, the pre-processing framework effectively reduce the runtime of the following global routing stage by around 50%.

During the experiments, I tried to use IOBL to differentiate characteristics between routable benchmarks and unroutable benchmarks. Other than "newblue3", I fail to observe significant difference bewteen the two types of benchmarks.

Table 4.3 Runtime Decomposition for APF

Name	IOBL	Auction	Global Routing
adaptec1	1%	45%	54%
adaptec2	7%	22%	71%
adaptec3	7%	40%	53%
adaptec4	21%	8%	70%
adaptec5	2%	24%	74%
bigblue1	1%	17%	82%
bigblue2	3%	19%	78%
bigblue3	7%	24%	68%
bigblue4	1%	19%	80%
newblue1	2%	13%	85%
newblue2	9%	9%	82%
newblue3	1%	9%	90%
newblue4	1%	12%	87%
newblue5	5%	14%	81%
newblue6	3%	16%	81%
newblue7	0.2%	5%	94.8%

CHAPTER 5. MGR: MULTI-LEVEL GLOBAL ROUTER

Even with FastRoute 4.0 and pre-processing detour framework, the sequential global router still lacks behind GRIP in total wirelength, which is a simple sum of metal wirelength and the number of vias. Decomposing the comparison, 2 stage sequential global routers actually is very good at congestion elimination and wirelength optimization. It is either the 2D nets has too many bends thus causing a lot of vias or layer assignment techniques are not functioning very well. Since many recent studies on layer assignment discover no major breakthrough, it is most likely that 2D router generates unsuitable routing patterns from the stand point of via reduction. As stated in the introduction, GRIP[5] provided an empirical lower bound of total wirelength but its practical value is limited. No design flow would spend hundreds of hours on global routing alone. The following work of parallel programming based GRIP [41] barely improves the practicality. The proposed usage of hundreds of distributed system is not reliable at all.

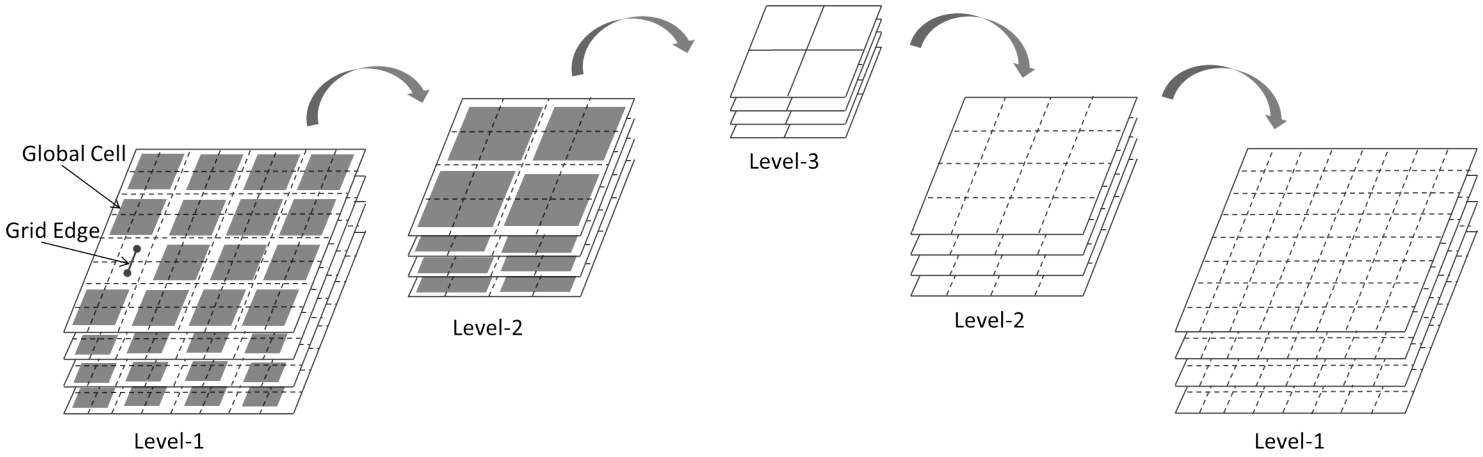
The only method to achieve good 3D global routing quality and short runtime seems to be 3D sequential global routing. But even the fastest 3D global router consumes five to ten times runtime comparing to its 2D counterparts to finish design. Experiments show there is not much room to improve 3D global routing runtime unless global routing adopts another framework.

So the third work of this thesis focuses on multi-level 3D global routing. It is the first work to introduce multi-level framework into global routers and is named MGR (Multi-level Global Router) accordingly. Rather than simply using multi-level framework to select nets to route in different levels, it proposes a novel resource reservation

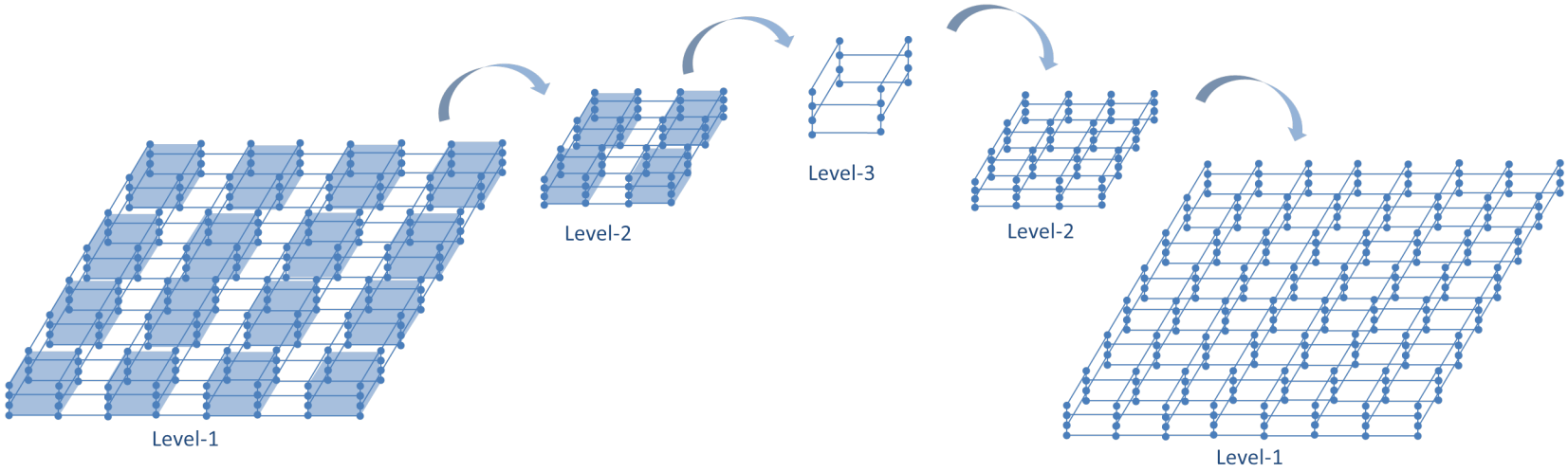
technique in coarsening stage to withhold routing resources form high level nets to get better routing solutions. More importantly, MGR uses concurrent ILP to propagate high level solutions to low level grid graphs. Such concurrency provides better solution than sequentially assigning high level nets to low levels. The sequential method is essentially equivalent to ordering long nets to short nets in ripup and reroute framework. The introduced concurrency considers the competition between different nets when they tries to use highly demanded wiring resources in congested region in the same time. As a results, MGR can generate solutions with quality much closer to GRIP only using runtime comparable to traditional framework with 2D global router and layer assignment.

5.1 Multi-Level Grid Graphs

For multi-level global routing, it is necessary to create multi-level grid graphs. The original cell structure and grid graph with finest structure is denoted as level-1. The level increases as global cells in cell structure become larger and grid graph becomes coarser. MGR derives the level-2 cell structure from level-1 cell structure in the following manner. 4 neighboring cells are merged into one, as shown in Fig. 5.1(a) by gray areas. The top level is denoted as level-t. Level-i grid graph models the crossing on boundaries of level-i cell structure. For level-(i+1) grid graph, the four corresponding level-i grid nodes grouped in shadow are merged into a node as shown in Fig. 5.1(b). One level-(i+1) grid edge corresponds to two level-i grid edges. The hierarchy is built up until each layer in the top level grid graph becomes smaller than or equal to the size of 8-by-8. In the hierarchy, nets that exist on level-i grid graph but disappear in level-(i+1) grid graph are denoted as level-i nets. Level-i nets do not belong to level-(i-1) nets because they still exist on level-i. Grid graphs at all levels have the same number of metal layers as level-1 grid graph and coarsening on different layers is vertically synchronized to maintain regular 3D cell and grid graph structures.



(a) Cell Structure Coarsening and Uncoarsening



(b) Grid Graph Coarsening and Uncoarsening

Figure 5.1 Cell Structure and Grid Graph Views of the Multi-Level Framework

5.2 Flow of MGR

The flow of MGR is illustrated in Fig. 5.2. MGR starts with 3D global routing initialization. MGR needs the initialization stage to provide guidance on how wire-length focused routing would generate congestion so it can detour high-level nets from the congested region. Our initialization consists of Steiner tree generation [30], a pattern routing step and dynamic programming based greedy layer assignment step just like previous 2D sequential routers [18] [24] without maze routing. Such process is very fast and can provide accurate enough congestion information for the following 3D multi-level rerouting stage.

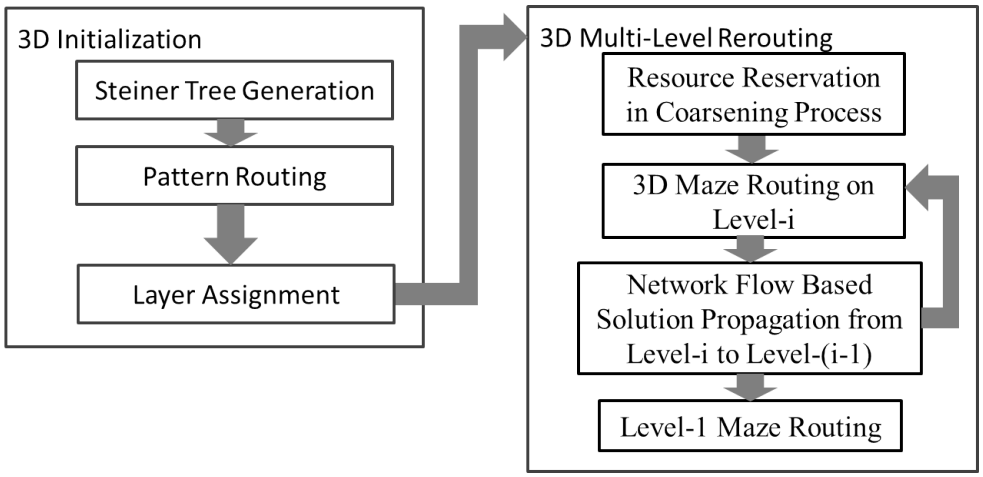


Figure 5.2 MGR Flow

In the rerouting stage, MGR uses 3D routing to explore the entire solution space and adopt multi-level framework to speed up such exploration. MGR first coarsens the grid graph to build hierarchy. The goal of coarsening is to generate higher level grid graphs to reduce the problem size and calculate accurate routing resource estimation based on the initial solution on those grid graphs. It uses adaptive resource reservation to adjust the capacities for high level grid edges, so that high level nets would not over utilize routing resources in congested region, which would reduce the need to rip-up and reroute high level routing solutions in low level grid graphs and thus effectively control run time

and improve solution quality. After the coarsening stage, MGR achieves all levels of grid graphs and their routing resource estimations. MGR then uses 3D maze routing, including the newly proposed 3-terminal maze routing, to eliminate congestion on the level-t grid graph. Then MGR uses network flow based solution propagation to project routing of level-t nets onto level-(t-1) grid graph. For a level-t net, its routing solution can only use grid edges on level-(t-1) grid graph its level-t solution represents. In general, before propagating level-i solutions to level-(i-1), MGR uses 3D maze routing to minimize congestion on level-i. Our newly proposed 3-terminal maze routing is adopted to enhance MGR's congestion reduction capability. This rerouting and propagation iteration inner loops as shown in the right part of Fig. 5.2 works from level-t to level-2. At the end of this loop, it achieves a rerouted level-1 solution. If the overflow remains after multi-level rerouting, MGR uses 3D maze routing on level-1 grid graph to eliminate it.

5.3 Multi-Level Global Rerouting Framework

5.3.1 Coarsening Process

The coarsening process generates smaller sized grid graphs for 3D maze routing. But it should not sacrifice routing quality with inaccurate estimations of routing resources on the coarsened grid graphs. MGR uses pattern routing and simple layer assignment to provide an initial solution so that the coarsening stage can calculate routing capability for higher level grid graphs based on the actual usage of lowest level grid graph. It differs from previous multi-level gridless routers in the way that previous works rely on inner cell routing blockage model to estimate boundary capacity.

In the coarsening process, MGR needs to make sure that it does not render high level grid graphs with excessive capacity so that high level nets lack incentive to avoid congested region. Such situation will naturally happen if MGR simply adds up the grid edge capacities of the two lower level grid edges corresponding to each higher level grid

edge. A higher level grid edge represents larger layout boundary. The possibility of overflow on that boundary tends to diminish as the level goes higher, due to the fact that the demand will average out over a long boundary. The larger area a cell includes, the less likely demands on its boundary exceed the capacity. Just like ideal global routing would detour global nets from locally congested global cells, it would be best for MGR to guide level-(i+1) nets to avoid congested edges on level-i. The most effective way to achieve this is through adaptively capacity adjustment for higher level edges. Without adjustment, the capacity of a level-(i+1) grid edge would be the sum of capacities of the two level-i grid edges it represents. This method has a major drawback. It ignores the congestion inside level-(i+1) cells and leads to much less detour than desirable amount. This stems from the nature of routing problem: If the boundary has ample routing resources while the cell is congested with intra-cell nets, routing on that boundary will aggravate intra-cell congestion. Thus, MGR adjusts the capacity of high level grid edges according to existing inner cell routing solution in an adaptive manner to guarantee that high level routing will not take up critical lower level routing resources for lower level nets.

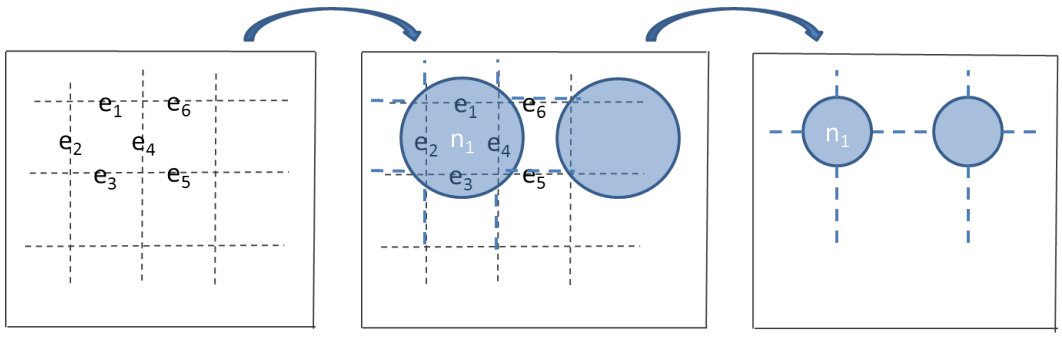


Figure 5.3 Coarsening Process in Grid Graph

Consider the coarsening step from level-i to level-(i+1), Fig. 5.3 shows a part of the grid graph that will go through the coarsening step. First, MGR contracts e_1, e_2, e_3, and e_4 into a level-i+1 grid graph node. For the node, MGR assigns a node adjustment value av_n, which is $o_{e1} + o_{e2} + o_{e3} + o_{e4}$, the sum of overflow of the level-i grid edges it includes.

The node adjustment value represents the need to discourage higher level nets to use the node in order to resolve congestion on level-i. Nevertheless, routing on a grid graph with both edge and node constraints overly complicates the problem. So in the second phase, MGR converts the node adjustment value to edge capacity adjustment on the same level by distributing the node adjustment value to the capacity of grid edges connecting to the node. For the level-(i+1) edges that connect to n_1, their capacities will each be reduced by a quarter of av_n. If the node is at the corner of grid graph, the capacities of the two edges connecting to the node will be reduced by half of the node adjustment value. In this way, MGR propagates level-i overflow information into level-(i+1) grid graph.

5.3.2 3D Maze Rerouting

During the uncoarsening process, MGR needs to reroute some nets to eliminate congestion. Recently, Min Et. Al. expanded maze routing to multi-source and multi-sink maze routing to handle multi-pin nets to further reduce congestion [32]. Instead of rerouting two subtrees from the two fixed endpoints of the ripped-up edge, they use any points on the subtrees as the reconnecting point to adaptively adjust net topology. However, the effectiveness of such adjustment is limited because it only optimally configures the one segment that connects the two unconnected subtrees while the subtrees themselves remain suboptimal. For multi-pin nets, which count for more than 40% in the benchmarks used in ISPD 2008 global routing contest [13], relying on 2-terminal maze routing to restructure net topology may lead global router to run many iterations before it can get a fairly good solution. Optimally connecting multi-pin nets will greatly improved solution quality. Our first effort goes to routing 3-pin nets optimally and it can be extended to optimize multi-pin nets iteratively.

On a second thought, rather than starting the analysis for rerouting 3-pin nets and later extending it to connecting subtrees, MGR draws lessons from [32] and propose an optimal method to connect nets ripped-up into 3 terminals directly. Here a terminal

could be a pin or a subtree. Generating the minimum cost solution to connect nets ripped-up into three terminals has two parts. While searching for a point to be the optimal Steiner point is the first obstacle, figuring out the paths between the Steiner point and each terminal stands as the second catch. One easy way to accomplish the two parts is through three independent wavefront propagations from the 3 terminals respectively. MGR can add the cost for the 3 propagation together and set the point with least sum to be the Steiner node. The optimal solution can be derived from back tracking the wavefront from the Steiner point to the terminals. The only problem is the efficiency of such technique. The search region has to be large enough to contain the optimal Steiner point, which means that the wavefront propagation might propagate to a large area. On the contrary, traditional 2-terminal maze routing can stop when the two wavefronts meet each other. Fortunately, it is easy to prove Theorem 1 to greatly limit the propagation area and thus improve the efficiency of the 3-terminal maze routing technique.

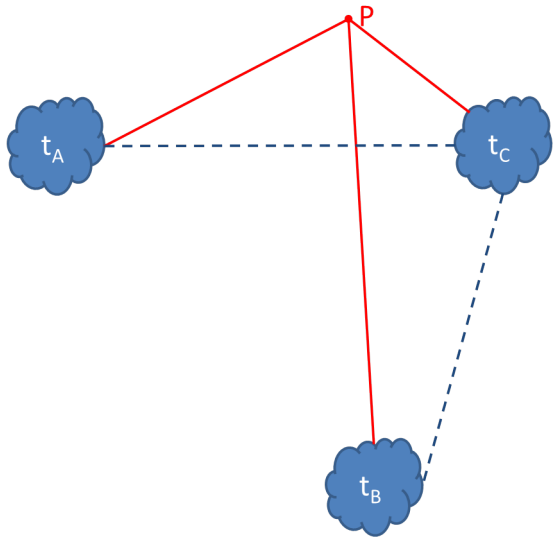


Figure 5.4 3 Terminal Maze Routing

Theorem 1. *The wavefront propagation from one terminal used in 3 terminal maze routing can stop whenever it reaches any of the two other terminals.*

Proof. As shown in Fig. 5.4, we have three terminals: t_A, t_B and t_C. Without loss of generality, we assume that we have an optimal Steiner point P, $md(t_B, P) > min(md(t_B, t_A), md(t_B, t_C))$. Here, "$md$" is the minimum path distance between two terminals. The inequality indicates that point P is out of range of the wavefront propagation started from terminal B. It is obvious that $md(t_A, P) + md(t_C, P) \geq md(t_A, t_C)$ due to triangular inequality. If we add the two inequalities together, we can get $md(t_A, P) + md(t_C, P) + md(t_B, P) > md(t_A, t_C) + min(md(t_B, t_A), md(t_B, t_C))$. The right part of the inequalities represents a solution that can connect the three terminals together with lower cost than the shortest paths through point P. Thus, we prove Theorem 1 by contradiction.

□

In the proof of Theorem 1, specify whether the terminal is a point or a sub-net is not specified. Thus, MGR can apply the multi-source multi-sink maze routing proposed in FastRoute 2.0 [32] to get a fast 3 terminal multi-source multi-sink maze routing. Besides, since minimum distance could be adjusted by any non-decreasing cost function, the algorithm works well for obstructions so long as routing over them gives large cost.

So the 3-terminal maze routing algorithm for nets with 3 pins or more runs as follows. Instead of ripping-up a 2-pin edge, MGR rip-ups three 2-pin edges that connect to a shared Steiner node, generating 3 separate terminals waiting to be connected. Then MGR starts the maze wavefront propagation for each terminal until it meets any one of the other two terminals. After the propagations, MGR finds the least cost Steiner node by checking the region visited by all three wavefront propagation procedure. With the least cost Steiner node at hand, the algorithm backtracks and constructs the paths to connect the 3 terminals together. The runtime bottleneck for 3-terminal maze routing still is the wavefront propagation, which results in a complexity of $O(n \cdot log(n))$, the same as traditional 2-terminal maze routing.

Traditionally, sequential routers run as little maze routing as possible to save runtime. One advantage for MGR is that it has a relatively small problem on higher level grid

graphs so it can afford running more maze routing. On higher levels, MGR picks up extra duty to balance out usage profile between large global cells to help eliminate lower level local congestions. To achieve that, MGR rips-up and reroutes every net using 3D maze routing for the top 2 levels to get better solutions. Running these extra maze routing improves high level solutions and consume little runtime due to small grid graphs and limited number of high level nets. On lower levels, MGR only use 3D maze routing to reroute congested nets, just like the behavior of traditional 2D sequential routers.

5.3.3 Routing Propagation to Lower Level

Once MGR reaches the top level and balances routing demands according to routing resources, it needs to propagate routing decisions made in the higher level to lower levels, in other words, from coarser grid graph to finer grid graphs.

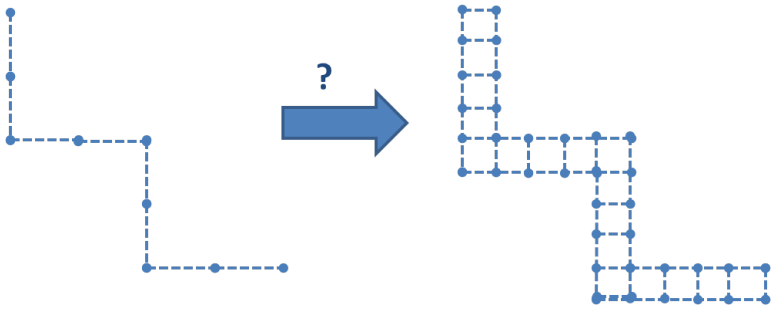


Figure 5.5 High Level Routing Propagation into Lower Level Grid Graph

The propagation for a single net is a relatively easy problem. As shown in Fig. 5.5, the higher level net just need to choose among the edges that is abstracted away during the corresponding coarsening process. Such propagation can easily be realized by dynamic programming. However, sequential net by net propagation ignores the impacts of one net onto its spatially correlated nets. The ideal solution with best results would be a simultaneous net propagation for all the high level nets. However, it has too big a problem size to finish in tight schedule, which is especially true for uncoarsening process for the few bottom levels. In the end, MGR comes up with a balanced solution between

solution quality and runtime concern. MGR propagates all the nets or part of the nets that use the same column or same row in the grid graph. Each column or row is called a slice. In a grid graph, a slice is defined as the grid nodes on all metal layers of same row or column index together with all the grid edges among the nodes.

Without loss of generality, let us pick a row on level-(i+1) for the following analysis. This row consists of 2 neighboring rows in level-i grid graph, together with all the vertical edges that connect them. This forms the backbone of one subproblem. In each subproblem, MGR uses network flow algorithm to propagate routing solution on the row into the finer lower level grid structure. To save more via, MGR relaxes the problem by allowing each high level net to choose new routing layer instead of the metal layer decided in its high level solution. This relaxation helps MGR to correct any layer assignment mistakes made in the previous level. The uncoarsening problem is decomposed into network flow sub-problems on slices. The decomposition disentangles the complex uncoarsening problems but still concurrently considers the competitions among all the nets that use a single slice.

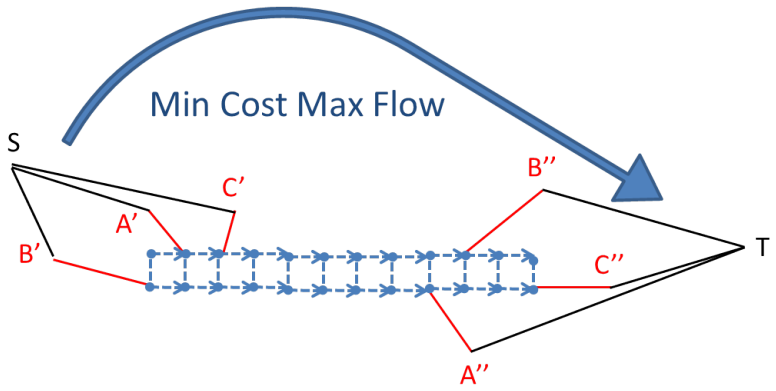


Figure 5.6 Network Flow for High Level Net Propagation

The formal problem formulation is defined as follows. MGR slices the entire column or row from the grid graph. On level-i, it has a single line with multiple layers while at level-(i-1), it has 2 grid-edge lines together with a row of vertical grid edges that connects them. This slice of grids forms the backbone structure of the network flow

problem. The grid graph edges are assigned a cost depending on the initial routing solutions and current routing solutions. The capacity and usage for the backbone is assigned in the following manner. Initially, the backbone has the capacity estimated during the coarsening process. Based on that, MGR adds fixed usage that represents the propagated routing solution by level-i propagation subproblems finished earlier, if any. Furthermore, for each level-i nets not propagated yet but will use the vertical grid edges in the backbone, MGR adds 0.5 usage on each of the two possible grid edges. For the network flow problem, MGR would compute the cost for each grid edge by a logistic function similar to [32] based on the capacity and usage calculated just above.

After the costs are calculated, MGR assigns net or part of net, which is denoted as a path from here, to the backbone to model routing demands on the slice. If a path has a pin on the slice, MGR creates an anchor at the pin location. Otherwise, if the path just "passes" the slice, MGR creates the anchor depending on the direction in which the path turns. Take net A in Fig. 5.6 as an example, it comes from up, use the the slice and goes down. So MGR creates anchor A' at the top row of the backbone and another anchor A" at the bottom. Every path that uses the slice has two anchors. If MGR scans through the slice from its minimum index to its maximum, whenever MGR finds the first anchor belonging to a path, MGR connects it to the source node "s". When MGR finds the second anchor, MGR connects it to the sink node "t". The scan adds edges to the backbone to model routing demands. Every edge connected to "s" or "t" has a capacity of 1 and a cost of 0. Besides, with the scan direction, MGR can assign the two rows of grid edges the same direction, which simplifies the network flow problem.

The optimal solution MGR wants to find is a minimum cost maximum flow from "s" to "t" on the network it just created. MGR use ILP to solve this discrete flow problem. Since the problem size is relatively small, comparing to the entire routing problem, runtime is not a concern here. If the maximum flow equals to the number of paths that has demands on the slice, the uncoarsening process for this slice is successful.

If not, then some high level nets cannot be routed. All the slices are sorted by how congested they are. The higher congestion one slice has, the earlier its network flow propagation would be solved, because less congested columns or rows could afford the reduced flexibility due to more fixed usage added onto their network flow backbone from earlier propagation subproblems.

MGR waits for the uncoarsening process for all slices to finish before it runs reroute for all unconnected high level nets, and some level-(i-1) nets if they are involved in congestion too. In the reroute process, MGR uses 3-terminal maze routing mentioned in previous section for nets with more than 2 pins and traditional maze routing for 2-pin nets. Then uncoarsening process will proceed to solve network flow based propagation problem from current level to the next lower level.

5.4 Experimental Results

MGR is implemented in C and conduct all the experiments on a Linux machine powered by a 2.6GHZ Intel Processor with 16GB memory. I still use the benchmarks from ISPD 08 global routing contest and compare the results of our work with leading academic global routers: NTHU-R 2.0 [18], NTUgr [23].

Table 5.1 Comparison of Routing Results on Routable Benchmarks for MGR

Name	MGR		NTHU-R 2.0 [18]		NTUgr [23]	
	Wirelength	cpu(s)	Wirelength	cpu(s)	Wirelength	cpu(s)
adaptec1	5282K	304	5344K	611	5740K	291
adaptec2	5146K	72	5229K	102	5370K	71
adaptec3	12892K	334	13101K	549	13500K	284
adaptec4	11996K	98	12169K	130	12370K	78
adaptec5	15323K	550	15538K	1160	15990K	988
newblue1	4558K	312	4653K	312	4930K	63161
newblue2	7446K	55	7570K	61	7690K	39
newblue5	22800K	453	23158K	977	24490K	1324
newblue6	17486K	487	17689K	912	18660K	1376
bigblue1	5582K	349	5595K	690	6000K	1169
bigblue2	8892K	415	9059K	427	9120K	16044
bigblue3	12875K	200	13068K	253	13350K	258
Comparison	1	1	1.015	1.7	1.053	23.5

Table 5.2 Comparison of Routing Results on Unroutable Benchmarks for MGR

Name	MGR			NTHU-R 2.0 [18]			NTUgr [23]		
	Overflow	WL	cpu(s)	Overflow	WL	cpu(s)	Overflow	WL	cpu(s)
bigblue4	134	22573K	1475	162	23090K	6633	188	24280K	26692
newblue3	31026	10722K	1384	31454	10653K	6168	31024	18830K	57120
newblue4	136	12854K	1083	138	13046K	4873	142	14380K	72246
newblue7	56	34902K	7624	62	35522K	7252	310	37220K	93401
Comparison	1	1	1	1.015	1.016	2.16	1.01	1.175	21.6

The comparison are conducted in two parts: for routable benchmarks and for unroutable benchmarks, separated by whether academic global routers can generate congestion free solutions up to now, because routers may adopt very distinctive behavior when facing the final few violations to resolve. Table 5.1 shows the comparison for routable benchmarks. Comparing to the contest winners, MGR generates solutions with least wirelength using much less run time. The solutions generated by MGR has 1.5%, 5.3% less wirelength comparing to NTHU-R 2.0, NTUgr respectively. In addition, the new routing framework runs $1.7X$ and $19.8X$ faster than the three global routers.

Table 5.2 shows the comparison for unroutable benchmarks. Once more, MGR generate solutions with shortest wirelength while using least runtime. The improvement for unroutable benchmarks is more significant than the improvement for routable benchmarks. It is because MGR could resolve the violations during the multi-level coarsening and uncoarsening process in a faster manner and later resort to level-1 3D maze routing to minimize the number of violations.

CHAPTER 6. SUMMARY AND DISCUSSION

This dissertation studies and works on physical design automation problems foglobal routing, which are summarized as follows:

Based on FastRoute framework, it proposes FastRoute 4.0, a very efficient 3D global router. FastRoute 4.0 takes via into consideration into nearly every aspects of global routing. In topology generation, it uses via-aware stiner tree generation technique to get routing structures that reduces via counts. In solution initialization stage, it uses 3-bend routing to replace monotonic routing to reduce congestion and via count. 3-bend routing is used in rip-up and reroute stage to partially replace maze routing to enhance speed. In the final stage, it uses dynamic programming based layer assignment technique to expand 2D global routing solutions into 3D ones. FastRoute 4.0 designs a framework that carefully integrates these via aware routing techniques together.

Furthermore, it proposes a detour pre-processing framework called APF for global routing. The framework contains a two step flow that first accurately identifies the most congested interval and later uses auction algorithm to simultaneously detour a minimum number of nets fully intersecting the interval to eliminate congestion. To improve the scalability, speed-up techniques are also proposed. The concept of most congested interval provides a reliable prediction of congested region in which routing has to detour. The auction algorithm based detouring algorithm provides an efficient way to concurrently detour nets with their interdependency in consideration. APF is the first work to provide helpful detouring information before any actual global routing and it could be directly be pluged into any global router with ease. The pre-processing

framework significantly improved the performance of FastRoute 4.0.

Last but not least, it continues the improvement of global routing solution quality by proposing a multi-level global router called MGR. MGRconsists of uncoarsening process and coarsening process. In the uncoarsening process, MGR adaptively adjusts the capacities for high level grid graphs to effectively direct high-level nets from congested region. In the coarsening process, we propose a novel three-terminal maze routing and network flow based solution propagation. These two new techniques together guarantee better routing solutions. This multi-level solution can greatly speeds up global routing and provides better solutions comparing to existing academic global routers.

The results outperform state-of-the-art global routers in wirelength, overflow and runtime. That being said, there is still quite a few areas to work on in the area of global routing. There is potential to model congestion hot spot in a more accurate manner, to introduce more concurrency to further optinmize routing solution without runtime overhead, or to improve current techniques to push routing capability even further. For the entire back end design flow, the room left for study is more abundant. With the complexity of design rules, global routing need to have a more close correlation with detail route to tackle design for manufatruability issues. Besides, we can integrate track assignment on upper metal layers into global routing to speed up the routing process and alleviate the burdern on detail routing.

BIBLIOGRAPHY

[1] http://www.itrs.net/

[2] C. Albrecht, "Provably good global routing by a new approximation algorithm for multicommodity flow", *Proc. of Int'l Symp. on Physical Design*, pp. 19-25, 2000.

[3] C. Albrecht, "Global routing by new approximation algorithms for multicommodity flow," *IEEE trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 5, pp. 622-631, 2001.

[4] M. Cho and D. Z. Pan, "BoxRouter: A new global router based on box expansion and progressive ILP," *Proc. of Design Automation Conference*, pp. 373-378, 2006.

[5] T. Wu, A. Davoodi and J. Linderoth, "GRIP: scalable 3D global routing using integer programming," *Proc. of Design Automation Conference*, pp. 320-325, 2009.

[6] R. Kastner, E. Bozorgzadeh and M. Sarrafzadeh, "Pattern Routing: Use and Theory for Increasing Predictability and Avoiding Coupling," *Proc. of IEEE transactions on Computer-Aided Design of Integrated Circuits and Systems*, VOL. 21, NO. 7, 2002.

[7] C.Y. Lee, "An Algorithm for path connections and its applications," *IRE Transactions on Electronic Computers*, VOL. EC-10, pp. 346-365, 1961

[8] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik, vol. 1*, pp. 269-271, 1959.

[9] P. E. Hart, N. J. Nilsson, B. Raphael, "Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *SIGART Newsletter*, vol. 37, pp. 2829, 1972.

[10] J. Hu and S. S. Sapatnekar, "A survey on multi-net global routing for integrated circuits," Integration, the VLSI Journal, vol. 31, no. 1, pp. 1-49, 2001.

[11] M. D. Moffitt, J. A. Roy and I. L. Markov, "The coming of age of (academic) global routing," invited paper *Proc. of Intl. Symp. on Physical Design*, pp 148-155, 2008.

[12] http://www.sigda.org/ispd2007/rcontest/.

[13] http:http://www.ispd.cc/contests/ispd08rc.html.

[14] R.T. Hadsell and P.H. Madden, "Improved global routing through congestion estimation," *Proc. of Design Automation Conference*, pp. 28-31, 2003.

[15] M. Cho, K. Lu, K. Yuan and D. Z. Pan, "BoxRouter 2.0: Architecture and Implementation of a hybrid and robust global router," *Proc. of Intl. Conf. on Compuer-Aided Design*, pp. 503-508, 2007.

[16] M. M. Ozdal and M. D.F. Wong, "ARCHER: a history-driven global routing algorithm," *Proc. of Intl. Conf. on Compuer-Aided Design*, pp. 481-487, 2007.

[17] J.-R. Gao, P.-C. Wu, and T.-C. Wang, "A new global router for modern designs," *Proc. Asia and South Pacific Design Automation Conf.*, 2008.

[18] Y.-J. Chang, Y.-T. Lee, and T.-C. Wang, "NTHU-Route 2.0: A Fast and Stable Global Router," *Proc. IEEE/ACM Intl. Conf. on Compuer-Aided Design*, 2008.

[19] J. A.Roy and I. L. Markov, "High-performance routing at the nanometer scale," *Proc. IEEE/ACM Intl. Conf. on Compuer-Aided Design*, pp. 496-502, 2007.

[20] L. McMurchie and C. Ebeling, "Pathfinder: A negoriation-based performance-driven router for FPGAs," *Proc. of Int'l Symp. on Field-Programmable Gate Arrays*, pp. 111-117, 1995.

[21] M. D. Moffitt, "MaizeRouter: Engineering an effective global router," *Proc. Asia and South Pacific Design Automation Conf.*, 2008.

[22] J. A. Roy and I. L. Markov, "High-performance Routing at the Nanometer Scale," *IEEE trans. on Computer-Aided Design of Integrated Circuits and Systems*, VOL. 27, NO. 6, pp. 1066-1077, 2008.

[23] Y. Chen, C. Hsu and Y. Chang"High-Performance Global Routing with Fast Overflow Reduction," *Proc. Asia and South Pacific Design Automation Conf.*, 2009, pp. 582-587.

[24] Y. Xu, Y. Zhang and C. Chu, "FastRoute 4.0: global router with efficient via minimization," *Proc. Asia and South Pacific Design Automation Conf.*, 2009, pp. 576-581.

[25] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, "Multi-Threaded Collusion-Aware Global Routing with Bounded-Length Maze Routing," *Proc. of Design Automation Conference*, pp. 200-205, 2010.

[26] Y. Zhang, Y. Xu and C. Chu, "FastRoute3.0: a fast and high quality global router based on virtual capacity," *Proc. of Intl. Conf. on Compuer-Aided Design*, pp. 344-349, 2008.

[27] J. Westra, C. Bartels and P. Groeneveld, "Probabilistic congestion prediction," *Proc. of Intl. Symp. on Physical Design*, pp 204-209, 2004.

[28] J. Westra and P. Groeneveld, "Is probabilistic congestion estimation worthwhile?" *Proc. Intl. Workshop on System-Level Interconnect Prediction(SLIP)*, pp. 99-106, 2005.

[29] Chris Chu, " FLUTE: Fast Lookup Table Based Wirelength Estimation Technique," *IEEE Intl. Conf. on Computer-Aided Design* , pp. 696-701, 2004

[30] C. Chu and Y. C.Wong, "FLUTE: Fast Lookup Table-based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 1, pp. 7083, 2008.

[31] M. Pan, and C. Chu, "FastRoute: A step to integrate global routing into placement," *Proc. of Intl. Conf. on Computer-Aided Design*, pp. 464-471, 2006.

[32] M. Pan and C. Chu, "FastRoute 2.0: A High-quality and Efficient Global Router," *Proc. Asia and South Pacific Design Automation Conf.*, pp. 250 - 255, 2007.

[33] M. Pan and C. Chu, "IPR: An Integrated Placement and Routing Algorithm," *Proc. of Design Automation Conf.*, pp. 59-62, 2007.

[34] T.-H. Lee and T.-C. Wang, "Robust Layer Assignment for Via Optimization in Multi-layer Global Routing," *Proc. International Symposium on Physical Design*, pp. 159-166, 2009.

[35] J. Cong, M. Xie and Y. Zhang, "MARS - A multilevel fullchip gridless routing system," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 3, pp. 382-394, 2005.

[36] T.-C. Chen, Y.-W. Chang, and S.-C. Lin, "A novel Framework for Multilevl Full-Chip Gridless Routing," *Proc. Asia and South Pacific Design Automation Conf.*, pp. 636 - 641, 2006.

[37] J. Westra, C. Bartels and P. Groeneveld, "Probablistic Congestion Prediction," *Proc. Int. Symp. on Physical Design*, pp. 204-209, 2004

[38] M. Bayati, D. Shah and M. Sharma, "A Simpler Max-Product Maximum Weight Matching Algorithm and the Auction Algorithm," *IEEE transactions on Information Theory*, VOL. 54, NO. 3, pp. 1241-1251, 2008.

[39] D. Bertsekas, "Auction algorithms for network flow problems: A tutorial introduction," *Computational Optimization and Applications*, VOL. 1, NO. 1, pp. 7-66, 1992.

[40] D. Bertsekas and J. Tsitsiklis, "Parallel and Distributed Computation: Numerical Methods," Prentice-Hall, Englewood Cliffs, J.J., 1989.

[41] T.-H. Wu, A. Davoodi, and J. T. Linderoth, "A parallel integer programming approach to global routing", Design Automation Conference, pp. 194-199, June 2010.

[42] M. Hanan, "On Steiner's problem with rectilinear distance," *SIAM Journal of Applied Mathematics*, vol. 14, pp. 255-265, 1966.