VFGR: A Very Fast Parallel Global Router with Accurate Congestion Modeling *

Zhongdong Qi[1], Yici Cai[1], Qiang Zhou[1], Zhuoyuan Li[2], Mike Chen[2]

[1] Tsinghua National Laboratory for Information Science and Technology,
Tsinghua University, Beijing, China
[2] Nimbus Automation Technologies, Shanghai, China
e-mail : zhongdongqi@gmail.com

Abstract - With the rapid growth of design size and complexity, global routing has always been a hard problem. Several new factors contribute to global routing congestion and can only be measured and optimized in 3-D global routing rather than 2-D routing. We propose an enhanced congestion model in global routing to capture local congestion and more accurately reflect modern design rule requirements. To achieve better global and detailed routing solution quality, we propose a 3-D global router VFGR with parallel computing using this congestion model. Experimental results show that VFGR can achieve comparable or better global routing solution quality with two start-of-the-art global routers in shorter runtime. It is also demonstrated that adopting proposed congestion model in global routing, higher solution quality and much shorter runtime can be achieved in detailed routing stage.

I. Introduction

Global routing is one of the most important stages in back-end VLSI design. It can serve as a congestion estimator to guide routability optimization in placement, and can also provide start point for detailed router to construct final layout. The chip's routability, timing, power and time-to-market are all heavily affected by global routing.

In global routing (GR), the routing region is tessellated into rectangular global routing cells (gcells), and a 2-D or 3-D routing graph is constructed to measure the routing congestion. In the graph, typically edge capacity is measured by the number of routing tracks between two gcells, while edge demand is the number of wire segments across the gcell boundary. Global router plans paths of nets on this graph. Guided by GR results, detailed routing (DR) searches and realizes real wires and vias inside all gcells.

As technology advances, several new factors are introduced, such as varying metal widths over the routing layers, via resource consumption and increase of design rules volume. They all enlarge the gap between GR congestion model and DR resource consumption situation, and contribute to complexity of global routing problem.

From 90nm technology node, varying width and thickness metal layers are available in VLSI design. At the 65nm technology node, typically three different metal widths, namely 1X, 2X and 4X are used across 9 metal layers (illustrated in Fig. 1). Even more metal widths and thickness are used across more metal layers in successive technology nodes. This makes routing truly a 3-D problem [1]. As shown in Fig. 1, varying metal widths and thicknesses introduce fat vias into interconnections, which consume much more resources than normal vias. In addition, more stacked vias are generated in global routing to form multi-layer connections as the number of routing layers increases. Resources consumed by fat vias and stacked vias can only be measured and optimized in 3-D global routing. Varying width metal layer stack and via resource consumption make 3-D global routing crucial.

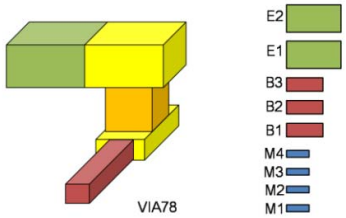

Fig. 1. A fat via and a metal layer stack with varying widths over different layers.

Due to sub-resolution lithography, chemical-mechanical polishing (CMP) and yield enhancement requirements, the number of design rules becomes larger in successive technology nodes. The design rules greatly affect amount of routing resources consumed by local connections and vias. Vias and local connections affected by design rules contribute a major mismatch between global routing congestion model and detailed routing resource consumption.

During global routing, an inaccurate congestion model misleads path search and enlarges the gap between global routing and detailed routing. The conventional congestion model cannot take above factors into account and underestimates resource consumption of global routes. Using the model, a global routing path could be unroutable in detailed routing stage, causing costly path rerouting in dense detailed routing grid. This makes a practical global congestion model essential.

In addition, 3-D global routing has a significantly larger search space than 2-D global routing, which would lead to much longer runtime. Today's multi-core architecture makes multi-threaded computing available, which can be used to accelerate this computational intensive problem.

In this paper, we propose an accurate and practical congestion model to capture some of resource consuming factors in modern technology. Using 3-D global routing graph based on the model, we develop a multi-threaded 3-D global routing algorithm named VFGR, which can produce high quality solutions using short runtime.

Our contributions are as follows:

1) An accurate congestion model capturing local congestion due to vias and local nets resource consumption along with

* This work is supported by National Natural Science Foundation of China (NSFC) No.61274031.

978-1-4799-2816-3/14/$31.00 ©2014 IEEE

related design rules. We demonstrate that using proposed congestion model in global routing can lead to great reduction of detailed routing runtime and the number of design rule violations.

2) A constructive hierarchical global routing framework and detailed techniques which are performed on 3-D routing graph. Experimental results demonstrate that good solution quality can be achieved.

3) Efficient parallelization of proposed global routing framework. A 6X speed up can be achieved using 8 threads on an 8-core CPU.

The rest of paper is organized as follows. Section II gives a brief statement of related work. Section III presents the details of proposed congestion model. The framework of VFGR and routing techniques are described in section IV, followed by experimental results. We conclude the paper in section VI.

II. Related Work

A. Congestion Modeling

In recent years, some pioneer works in global routing or routability estimation were done to remedy the inconsistency between GR congestion model and DR resource consumption. Hsu et al. [4] proposed concepts of via capacity and via overflow to take stacked vias into account. As the technology node becomes smaller, this model is not practical and out dated. After that, in work of Taghavi et al. [21], pin geometries and density was used to measure detailed routing difficulty in routability-driven placement. Wei et al. [23] used pin density factor to measure local congestion. Local resource usage is considered in by increasing edge demand. Most recently, Shojaei et al. [20] used vertex demand to measure local nets resource consumption.

The above works improved the accuracy of routability estimation during placement or congestion modeling in global routing. However, fat vias were not modeled, and design rules were not modeled explicitly. The vias and local nets were not measured in a uniform model.

B. Global Routers & Parallel Global Routing

Significant progress on global routing algorithms has been made in recent years. A number of high quality global routers were developed, which can be roughly categorized into two classes, sequential ones and concurrent ones, according to how the nets are routed.

In sequential class, FGR [19], MaizeRouter [16], Archer [17], NTHU-Route 2.0 [3], NTUgr [4], FastRoute [18], MGR [26], BFG-R [11], NCTU-GR 2.0 [14] and several other high performance global routers use ripping-up and rerouting technique to get overflow minimized solutions. Among these routers, FGR, BFG-R and MGR can perform 3-D routing directly, while others comprise of 2-D global routing and layer assignment. In concurrent class, routers typically use mathematical programming methods to route a group of nets at the same time. BoxRouter [5] and GRIP [24] use integer programming to find routes for nets in a routing region.

There has been very few works on parallel global routing to date. PGR [15] introduces a net-level collision concept to multi-threaded global routing. PGRIP [25] uses region partition to generate sub-problems which solved using integer programming in a distributed computing cluster. Although PGRIP is parallel, the usage of integer programming makes the runtime still relatively long. The work in [10] adds computing with GPU into parallel global routing. In this paper, we try to explore region-level and net-level parallelism in shorter runtime using multi-threaded computing.

III. Accurate Congestion Modeling

The explosion of design rules makes modeling all the rules impossible. Many rules are seldom triggered in practical design. Two major design rules affecting local congestion are minimum area rule and end-of-line spacing rule (depicted in Fig. 2). Minimum Area Rule (*MinArea*) specifies the minimum metal area required for polygons on the layer. All polygons must have an area that is greater than or equal to a minimum area value. End-of-Line Spacing Rule (*EOL-Spacing*) indicates that a polygon edge that is shorter than certain length requires spacing greater than or equal to a spacing value beyond the end-of-line within some distance.

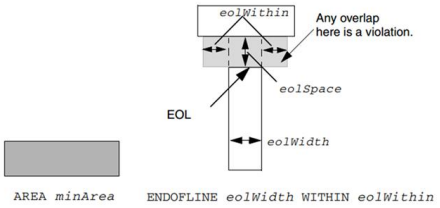

Fig. 2. Illustration of MinArea and EOL Spacing rules [13]

To practically model detailed routing resource consumption, let's first look at an example of a gcell with detailed routing results shown in Fig. 3. This gcell has 8 wire segments and two fat vias (one stacked and one non-stacked) in it. For inter-gcell connections, it has 7 and 6 wire segments on left and right gcell boundary respectively, resulting edge congestion as 7/12 and 6/12 on corresponding left and right global routing graph edges. Measured by these values, the gcell is not congested. However, considering intra-gcell connections and design rules, the region is too congested to let an extra wire segment cross the gcell. The fat via enclosure's width is wider than the minimal width on the layer. Considering the spacing rule, adjacent two tracks are not available for segments to pass through. As for a stacked via enclosure, its area is smaller than the minimal area value in MinArea rule, so the enclosure is enlarged with an extra wire to obey the rule.

From this case, we can see that (1): the fat vias and stacked vias can cause serious routability problems in detailed routing if they are not treated carefully at global routing stage; (local nets contribute to congestion in a similar way) (2): only inter-gcell congestion (edge congestion in global routing grid graph) is not capable to measure real detailed routing congestion. So we use pass-through capacity and demand to measure intra-gcell congestion, which is embedded in vertices of 3-D global routing graph. A simple example is depicted in Fig. 4.

In this graph, inter-gcell congestion is measured by edge capacity and demand, and intra-gcell congestion is measured by pass-through capacity and demand.

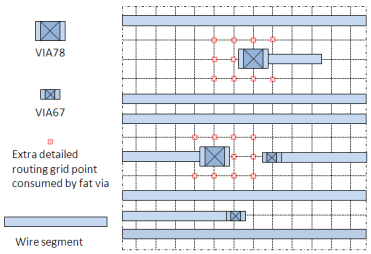

Fig. 3. A gcell with 2 fat via enclosures and 8 wire segments.

To avoid design rules checking surprise in detailed routing stage, we need to assign enough amounts of resources to each feature of net routes in global routing.

A **fat via** enclosure on thin metal layer is as wide as minimum width of the upper thick metal layer which it connects. For example, in a typical 65*nm* technology, due to minimum spacing rule on thin metal layer, the via enclosure makes neighboring two wiring tracks unavailable for placing wire segments to pass through. So a fat via contributes 3 to pass-through demand of this gcell on thin metal layer.

A **stacked via** have enclosures on connected metal layers. An enclosure may trigger MinArea and EOL-Spacing rule. To avoid violations, the enclosure is enlarged to *minArea* with spacing *eolSpace* on both ends. As a results, for a stacked via enclosure, the contributed pass-through demand is *minArea* / *enclosure_width* + 2 * *eolSpace*.

A **local net** (or subnet of a global net) has its pins all in one gcell. In a detailed routed design, a local net is typically routed using Metal2 and Metal3. Each local net is decomposed to segments using rectilinear minimum spanning tree construction. Since a local net is relatively small, a wire segment can easily violate MinArea and EOL-Spacing rules. The way of computing pass-through demand contributed by a local net segment is similar to that of a stacked via enclosure.

Inter-gcell wire segments also contribute to pass-through demand. A wire passing through the gcell increases pass-through demand by 1. The wire segments connecting to the gcell but not passing through it (denoted as side wire segments) contribute $\max\{N_l, N_r\}$ to pass-through demand, where N_l and N_r are number of left/lower and right/upper side wire segments connecting to the gcell, respectively. For example, $N_l = 2$ and $N_r = 1$ for the gcell in Fig. 3.

This congestion model is compatible with widely-used path search algorithms in global routing, such as pattern routing and maze routing. Conventionally, these path search algorithms are carried out on 2-D/3-D global routing grid graph edges. The cost function is generally related to edge congestion, wirelength, via count etc. Since vertices are connected by edges in the routing graph, it is natural to perform these algorithms on the routing graph to utilize both edge and vertex information. The only revision for the path search algorithms is to take vertex (pass-through) congestion into account.

IV. Multi-threaded 3-D Global Routing

A. *Proposed Global Routing Framework*

In global routing problem, nets compete for limited routing resources to achieve their own routes. In our observation on designs routed by academic and commercial routers,

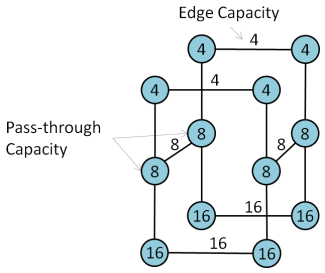

Fig. 4. Proposed congestion model in 3-D global routing graph.

characteristics of the resources consumed by a net are highly correlated with the size of the net. Generally, smaller nets consume resources in smaller region on lower routing layers, and larger nets take resources in a larger scope on higher routing layer. Smaller net generally has less routing flexibility, while larger net has more. This principle was used in several previous works [2] [7].

From this observation of resource distribution characteristics, we propose a hierarchical 3-D global routing approach. In this approach, multiple levels of hierarchy are constructed on the routing region, in which each hierarchy level has different size of global routing cell. Nets are fitted into different hierarchy level according to their bounding box size. Global routing is carried out from bottom hierarchical level to top level. In each hierarchy level, parallelism in region-level or in net-level is used, which is presented in subsection IV-C.

The design flow of proposed global routing is depicted in Fig. 5. Detailed techniques employed in the flow as well as parallelization of different level routing are presented in the following sub-sections.

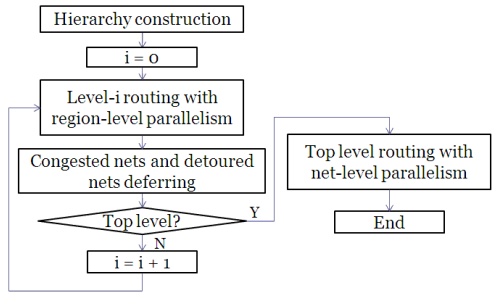

Fig. 5. Flow of proposed global routing.

For hierarchical global routing, hierarchical 3-D grid graphs are constructed beyond original global routing grid graph. We denote the original global routing grid graph as level 0 graph. Each level-(i+1) graph is constructed by merging level-i graph neighboring N x N gcells (N is 4 in our implementation). The graph coarsening procedure stops when current level coarse graph has row or column number less than predetermined threshold.

All the nets are decomposed into 2-pin subnets (as described in subsection IV-B1), and are fitted to each hierarchy level according to bounding box size. A subnet is mapped to level-i if its bounding box is not larger than level-i graph gcell but is larger than level-(i-1) graph gcell. Nets in each hierarchy level include nets fully embedded in a gcell (denoted as gcell nets) and nets across gcell boundary (denoted as boundary nets).

The hierarchical global routing proceeds from bottom level

to top level. In each level routing, gcell nets are routed in region of one gcell. Carried out in each region of 2 x 2 gcells in current level graph, boundary nets are then routed. Because the boundary nets cross different boundaries with even and odd indices, four passes of 2 x 2 gcell routing is performed with different start row and start column index.

In each regional routing problem, 3-D path search is directly performed on original global rouging grid graph (details are in subsection IV-B2). We do not utilize coarsened graph to route nets due to difficulty of modeling congestion on the coarsened graph.

After routing in each region, congested nets and detoured nets are deferred to higher level routing, which is presented in subsection IV-B3.

B. Detailed Techniques

(1). Flexible Net Topology and Resource Sharing

In global routing, net decomposition method, which impacts net tree topology, has strong impact on global routing solution quality and runtime. Rectilinear Steiner minimal trees (RSMT) and rectilinear minimum spanning tree (RMST) are two choices for decomposition. RSMT has shorter wirelength but less flexibility for rerouting, while RMST has better flexibility for rerouting but relatively longer wirelength. In our router, both RSMT and RMST are utilized to deliver advantages of both kinds of net topology. We use minimum spanning tree (MST) to decompose each multiple pin net to 2-pin subnets, and use RMST generated by FLUTE [6] to guide the path search of each subnet. Guided by RMST, a path search has zero wirelength cost for an edge connected to a Steiner point. This helps wirelength reduction during subnet routing. An example is shown in Fig. 6.

A necessary partner technique with net decomposition using MST is resource sharing between subnets of each net [19]. In global routing grid graph, each edge records which nets are utilizing it. A path search has zero wirelength cost using an edge which is already utilized by another subnet of the net.

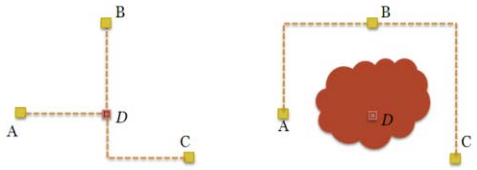

Fig. 6. A net with three pins *A*, *B*, and *C*. Point *D* is Steiner point. The net is decomposed into two subnets *A*-*B* and *B*-*C*. If there is no congestion around Steiner point *D*, paths passing *D* is created for both subnets, with resource sharing between them. If congestion exists in region around *D*, overflow-free paths can also be created.

(2). Regional Routing with Bounded Box Path Search

In each hierarchical level, routing of nets in each sub-region is proceeded using ripping-up and rerouting (RRR). Specifically, negotiated-congestion routing [27] is used. Each subnet is rerouted by a 3-D A* search engine. The iterative RRR finishes when overflow is reduced to zero or iteration count reaches a threshold.

An important aspect of hierarchical routing is to ensure appropriate resource consumption by nets in different levels. In regional routing of a certain level, routing path of a subnet should be constrained into a bounded box. If a subnet path occupies much bigger region than bounding box of its pins, it consumes resources belonged to remaining unrouted nets. In order to limit resource usage, A* search is performed in a bounded box to facilitate bounding box control of routed path. During several rounds of rerouting, bounding box of A* search is gradually enlarged to find a less congested feasible solution.

(3). Deferring Congested and Detoured Nets to Higher Level

In regional routing, a net path could be congested after iterative RRR, which implies current region restricts the path search of the net. A net path could also be much longer than its optimal path wirelength (an example is shown in Fig. 7), which consumes the resources of other unrouted nets. These nets should be routed in a larger region. When routing in a region finishes, congested nets as well as nets with detour ratio larger than a threshold are deferred to higher level routing and rerouted in higher level. This enables a more natural resource utilization.

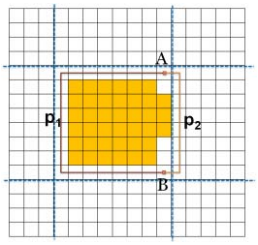

Fig. 7. A subnet *A*-*B* with long detoured path p_1. Blue dashed lines are boundaries of gcells on this level. Black solid lines are boundaries of gcells of original routing graph (level-0 gcells). Orange colored gcells are 100% full. It's proper to have path p_2 for the subnet, which can be achieved by deferring this subnet to higher level routing.

C. Parallelism in Different Level Routing

We now present method of parallelization in different level routing, using multi-threaded computing, in shared memory multiple-instruction multiple-data (MIMD) computing platform.

In proposed algorithm framework, the most time consuming part is regional routing in each level. Because all regions are independent in data, i.e. subnets to be routed and portion of global routing grid graph used in regional routing, *region-level* parallelism can be performed to speed up entire global routing.

However, higher levels which contain relatively small amount of gcells could have poor load balance during routing. If the amount of regions to perform routing is less than the amount of available CPU cores, some cores are inevitably idle while other cores are working. Similar situation occurs when number of nets to be routed varies greatly over different regions and amount of regions is close to the amount of CPU cores.

To handle this issue, we utilize *net-level* parallelization for higher level routing. During hierarchy construction phase of algorithm, graph coarsening is set to stop when row or column number of next level coarse graph is less than 2 * *K*, where *K* is the amount of cores. The reason of choosing 2 * *K* is that boundary nets are routed in a region with size of 2*2 gcells. This level is set to be top level of hierarchical routing, and contains all remaining unmapped nets. These nets are routed

in parallel in a path search bounding-box overlap free manner.

(1). Region-level parallelism in lower level routing

For a lower hierarchical level, routing of each region is constructed as a computing task. The routing procedure presented in subsection IV-B2 needs no revision. A routing space containing a partial global routing grid graph covered by the region, and all subnets to be routed in the region are prepared as task data. All the tasks are pushed into task queue and are consumed by available CPU cores. When a CPU core finishes a task, it obtains a new task from task queue.

(2). Net-level parallelism in top level routing

In top level, routing of each subnet is a computing task. A pointer to common routing space containing the entire global routing grid graph, and pins of the subnet are constructed as task data. All the tasks are constructed are pushed into task queue.

Routing procedure in each subnet task is to gradually enlarge path search bounding box (BBox) and search for a path using bounded box A* search engine. Each subnet in routing task has a record of a *path search BBox* (not subnet BBox) for current path search. To avoid multiple threads access or modify a global routing grid graph edge or vertex congestion information in the same time, all the path search BBoxes of tasks proceeded in a certain time are guaranteed to have no overlap. This is achieved by judging a task's path search BBox. If a task has a path search BBox not overlapped with those of current processing tasks, it's compatible with current tasks and will be consumed by a thread, else it will be pushed back to the task queue.

V. Experimental Results

We implemented VFGR using C++. The C++ code is compiled and run on a server with Intel 8-core 2.40GHz CPU and 24GB memory.

The benchmarks we used are the DAC 2012 placement and routing benchmark suite which contains several challenges in modern technology [22]. These designs contains massive hard macros, varying metal layers width and spacing, pins on higher metal layers. The placement solutions we used are produced by the winner placer NTUplace4 in DAC 2012 routability-driven placement contest (achieved from [8]).

Two flows using different settings of the benchmark suite are performed to demonstrate effectiveness of our global router and proposed congestion model respectively (subsection V-A and V-B). The first one uses the original gcell size and routing capacity setting in DAC 2012 benchmark suite to only perform global routing. The second one uses modified smaller gcell size and routing capacity to run full-flow global routing and detailed routing. The details are as follows. To facilitate detailed routing, we change the gcell size from 32x40 to 9x9 (9 is the placement row site height, in unit of 1 pitch) in second setting. We map one unit in placement image to one pitch (i.e. 180*nm* in 65*nm* technology node) in Metal1. We also add the 65*nm* design rules to the designs and construct the pin figures and obstacles in standard cells. The benchmark files are transformed to LEF and DEF files and also OpenAccess database. This set of benchmarks can be used for both global routing and detailed routing.

A. Performance of VFGR

To demonstrate the performance of our global router, we compare the runtime and solution quality of BFG-R and NCTU-GR 2.0 with that of VFGR under traditional congestion model (by turning off vertex capacity and design rules related congestion factors) on original DAC 2012 benchmarks.

BFG-R runs in multi-layer mode and performs 3-D routing. NCTU-GR 2.0 comprises of 2-D global routing and layer assignment, and run in regular mode. Since the released NCTU-GR 2.0 version does not support multi-threaded computing, it is performed using a single CPU core. VFGR does parallel routing using 8 threads. Three routers run on the same machine with an 8-core CPU.

The results are listed in Table 1. "OF" represents number of total edge overflow. "WL" is routed wirelength, which is in unit of 10^{10} *nm*. "via" is total via count in unit of 10^6. "E-CPU" denotes elapsed runtime of global routing, in unit of minutes, while "CPU" is the total CPU time of 8 cores.

As a 3-D global router, VFGR is 1.24x and 7.29x faster than another 3-D router BFG-R in term of CPU time and elapsed time, respectively. Since NCTU-GR 2.0 uses 2-D routing followed by layer assignment, which has smaller search space than 3-D router, NCTU-GR 2.0 spends shorter CPU time than VFGR. Using multi-threaded computing, VFGR is faster than NCTU-GR 2.0 measured by elapsed runtime. In term of overflow, wirelength and via count, VFGR achieves comparable or better solution quality with NCTU-GR 2.0 and BFG-R.

The speed up of parallel computing in VFGR is about 6X on average, which is relatively high for a multi-threaded program run on 8-core CPU. This implies data dependency in global routing is controlled to a limited scope using proposed hierarchical global routing framework.

B. Effectiveness of Proposed Congestion Model

We use two full-flow routing configurations to show the influence of different congestion models. In these two configurations, global routing solutions are achieved by VFGR with proposed congestion model and BFG-R, respectively. Then a commercial detailed router supporting 130*nm* to 45*nm* design rules is used to perform detailed routing. The detailed routing results are checked by a commercial design rule checker. BFG-R uses traditional congestion model only considering wire congestion. The benchmark suite is the one with smaller gcell size to adapt with the detailed router. We compare the results of global routing and detailed routing phases which are listed in Table 2. "OF" represents number of total edge overflow (both edge and vertex overflow for VFGR). "WL" is routed wirelength, which is in unit of 10^{10} *nm*. "via" is total via count in unit of 10^6. "CPU" denotes runtime in unit of minutes. "DRC" is total number of design rule violations, in unit of 10^3.

Using proposed congestion model, VFGR produced slightly better wirelength and vias in global routing stage than BFG-R, with much shorter elapsed runtime. Guided by VFGR solutions, detailed router produced 59% fewer DRC violations and 6% and 9% smaller wirelength and via count in detailed routing solutions with 51% shorter runtime compared to the one using BFG-R global routing results. VFGR reported many edge and pass-through overflows which indicated detailed routing congestion (i.e. DRC violations).

These results validate the effectiveness of proposed global congestion model, which is better correlated to detailed routing. This indicates that proposed congestion model using both edge and vertex capacity captures detailed routing congestion much more accurately than conventional model does. The global routing paths guided by proposed model can guide detailed router finding feasible interconnections with less design rule violations and avoids massive rerouting in dense detailed routing grid.

VI. Conclusion

Facing the challenges brought by successive technology nodes for global routing, we propose enhancements to current congestion models, to capture local congestion and more accurately reflect modern design rule requirements. We propose a hierarchical 3-D global router with parallel computing named VFGR. Experimental results show the effectiveness of proposed parallel global routing algorithm. It is also demonstrated that proposed congestion model is better correlated to detailed routing resource consumption than conventional model.

References

[1] Charles J. Alpert, et al. "What makes a design difficult to route." in Proc. of ISPD, 2010, pp. 7 - 12.

[2] Yao-Wen Chang and Shih-Ping Lin. "MR: a new framework for multilevel full-chip routing." IEEE TCAD 23.5 (2004): 793 - 800.

[3] Yen-Jung Chang, et al. "NTHU-Route 2.0: a robust global router for modern designs." IEEE TCAD 29.12 (2010): 1931 - 1944.

[4] Chin-Hsiung Hsu, Huang-Yu Chen, and Yao-Wen Chang. "Multi-layer global routing considering via and wire capacities." in Proc. of ICCAD, 2008, pp. 350 - 355.

[5] Minsik Cho and David Z. Pan. "BoxRouter: a new global router based on box expansion and progressive ILP." IEEE TCAD 26.12 (2007): 2130 - 2143.

[6] Chris Chu and Yiu-Chung Wong. "FLUTE: Fast lookup table based rectilinear steiner minimal tree algorithm for VLSI design." IEEE TCAD 27.1 (2008): 70 - 83.

[7] Jason Cong, Jie Fang, and Yan Zhang. "Multilevel approach to full-chip gridless routing." in Proc. of ICCAD, 2001, pp. 396 - 403.

[8] http://archive.sigda.org/dac2012/contest/dac2012_contest.html

[9] Ke-Ren Dai, Wen-Hao Liu, and Yih-Lang Li. "NCTU-GR: efficient simulated evolution-based rerouting and congestion-relaxed layer assignment on 3-D global routing." IEEE TVLSI 20.3 (2012): 459 - 472.

[10] Yiding Han, et al. "Exploring high throughput computing paradigm for global routing." in Proc. of ICCAD, 2011, pp. 298 - 305.

[11] Jin Hu, Jarrod A. Roy, and Igor L. Markov. "Completing high-quality global routes." in Proc. of ISPD, 2010, pp. 35 - 41.

[12] Ryan Kastner, et al. "Pattern routing: use and theory for increasing predictability and avoiding coupling." IEEE TCAD 21.7 (2002): 777 - 790

[13] LEF/DEF reference 5.7, Cadence, Nov. 2009

[14] Wen-Hao Liu, et al. "NCTU-GR 2.0: Multithreaded Collision-Aware Global Routing With Bounded-Length Maze Routing." IEEE TCAD 32.5 (2013): 709 - 722.

[15] Wen-Hao Liu, et al. "Multi-threaded collision-aware global routing with bounded-length maze routing." in Proc. of DAC, 2010, pp. 200 - 205.

[16] Michael D. Moffitt. "MaizeRouter: Engineering an Effective Global Router." IEEE TCAD 27.11 (2008): 2017 - 2026.

[17] Muhammet M. Ozdal and Martin D.F. Wong. "Archer: a history-based global routing algorithm." IEEE TCAD 28.4 (2009): 528 - 540.

[18] Min Pan, et al. "FastRoute: an efficient and high-quality global router." VLSI Design 2012 (2012): 14.

[19] Jarrod A. Roy and Igor L. Markov. "High-performance routing at the nanometer scale." IEEE TCAD 27.6 (2008): 1066 - 1077.

[20] Hamid Shojaei, Azadeh Davoodi, and Jeffrey Linderoth. "Planning for local net congestion in global routing." in Proc. of ISPD, 2013, pp. 85-92.

[21] Taraneh Taghavi, et al. "New placement prediction and mitigation techniques for local routing congestion." in Proc. of ICCAD, 2010, pp. 621 - 624.

[22] Natarajan Viswanathan, et al. "The DAC 2012 routability-driven placement contest and benchmark suite." in Proc. of DAC, 2012, pp. 774 - 782.

[23] Yaoguang Wei, et al. "GLARE: global and local wiring aware routability evaluation." in Proc. of DAC, 2012, pp. 768 - 773.

[24] Tai-Hsuan Wu, Azadeh Davoodi, and Jeffrey T. Linderoth. "GRIP: Global routing via integer programming." IEEE TCAD 30.1 (2011): 72 - 84.

[25] Tai-Hsuan Wu, Azadeh Davoodi, and Jeffrey T. Linderoth. "A parallel integer programming approach to global routing." in Proc. of DAC, 2010, pp. 194 - 199.

[26] Yue Xu and Chris Chu. "MGR: Multi-level global router." in Proc. of ICCAD, 2010, pp. 250 - 255.

[27] Larry McMurchie and Carl Ebeling, “PathFinder: a negotiation-based performance-driven router for FPGAs,” in Proc. of ACM Int. Symp. on FPGAs, pp. 111–117, 1995.

TABLE I. Global routing results comparison under conventional congestion model.

	VFGR					NCTU-GR 2.0				BFG-R			
Testcase	OF	WL	via	E-CPU	CPU	OF	WL	via	CPU	OF	WL	via	CPU
superblue2	0	9.32	5.25	6.93	45.07	0	9.21	5.29	16.40	0	9.61	5.35	97.48
superblue3	0	5.5	4.82	5.48	33.98	0	5.42	4.81	11.05	0	5.99	5.17	49.83
superblue6	0	5.36	4.93	3.23	19.40	0	5.3	4.94	6.55	0	5.54	5.21	19.60
superblue7	0	6.48	7.29	3.87	22.42	0	6.46	7.31	5.90	0	7.33	7.92	25.32
superblue9	0	3.95	3.96	2.17	12.35	0	3.91	4.01	4.23	0	4.40	4.23	13.65
superblue11	0	5.22	4.2	2.15	12.03	0	5.21	4.37	2.82	0	5.49	4.37	14.68
superblue12	0	5.55	6.71	2.20	12.53	0	5.49	6.68	4.08	0	6.05	6.79	11.67
superblue14	0	3.54	3.32	3.50	17.85	0	3.51	3.30	3.22	0	3.69	3.61	18.48
superblue19	0	2.34	2.22	1.02	5.50	0	2.35	2.29	1.13	0	2.37	2.40	6.12
Average	0	1.000	1.000	0.17	1.00	0	0.992	1.009	0.29	0	1.066	1.058	1.24

TABLE II. Full-flow routing results using different global routers.

	BFG-R								VFGR with Proposed Congestion Model							
Stage	GR stage				DR stage				GR stage				DR stage			
Testcase	OF	WL	via	CPU	DRC	WL	via	CPU	OF	WL	via	E-CPU	DRC	WL	via	CPU
superblue2	0	10.01	7.58	306.3	2355	11.4	12.54	484.8	2056	9.92	7.51	39.5	713	10.7	11.5	271.5
superblue3	0	5.77	7.28	137.9	1312	6.69	11.45	516.2	2408	5.75	7.13	16.1	602	6.24	10.9	242.6
superblue6	0	5.54	7.42	122.7	1026	6.32	9.39	351.6	1516	5.49	7.27	17.9	513	5.93	9.12	179.3
superblue7	0	7.48	11.0	82.6	685	8.48	15.61	392.8	994	7.25	10.8	9.1	305	7.92	14.3	157.1
superblue9	0	4.39	6.31	51.7	316	5.13	10.58	308.9	330	4.34	6.19	6.5	116	4.77	9.53	139.0
superblue11	0	5.58	6.61	100.2	2561	6.41	10.21	329.6	4708	5.51	6.60	11.8	1256	6.01	9.91	168.1
superblue12	0	6.15	10.7	69.9	812	7.11	14.70	251.2	1302	6.10	10.5	7.6	319	6.62	14.0	145.7
superblue14	0	3.71	4.72	74.0	5910	4.29	8.40	340.2	10716	3.69	4.61	9.2	2294	4.01	7.5	170.1
superblue19	0	2.43	3.55	31.5	6627	2.73	7.76	279.6	9407	2.42	2.51	5.9	2198	2.64	5.69	136.7
Average	0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	3715	0.99	0.98	0.13	0.41	0.94	0.91	0.49