OGRE: Open-Source Global Router

Habiba Gamal, Ali El-Said, Fady Mohamed, and Mohamed Shalan
The American University in Cairo, New Cairo, EGYPT

***Abstract*—In this work, we present OGRE, an open source LEF/DEF global router that produces guide file following the ISPD-2018/2019 format. OGRE works on a 3D grid and uses A* maze routing algorithm, with an added cost function that makes the routing via-aware and congestion-aware. OGRE is built out of publicly-available open source libraries and components by a group of undergraduate students as a course project. OGRE is tested against ICCAD 2019 global router benchmark designs and found to produce high quality valid output when compared to other academic global routers.**

Index Terms*—VLSI, EDA, GCell, Global Routing, Physical Design Automation, Maze Routing, SALT, A

I. Introduction

In the physical design of very large scale integrated circuits (VLSI), routing is an important design step that affects the circuit timing, power consumption, chip reliability and manufacturability, among other metrics. The criticality of the routing problem is increasing now more than ever because of the increase in the complexity of the designs which in turn increases the chip density, routing demand and risk of routing failure. Moreover, the feature size is decreasing which calls for more complex design rules to ensure manufacturability, making it harder to generate design rule checking (DRC) free routing solutions [1]. This leads to breaking down the routing into two steps: global routing followed by detailed routing.

A global router divides the 3D routing grid into rectangular routing regions called global cells (GCells). Each edge in this grid is assigned routing capacities. A global router guides the detailed router through finding a Gcell-to-Gcell path while meeting the specified constraints [2]. Among these constraints are: avoiding overflow which is exceeding the routing capacity of a global cell (Gcell), reducing congestion which happens when the used routing resources of a Gcell is close to the maximum capacity and reducing total wirelength. Moreover, vias are the major reason for circuit failures during manufacture [3]. Therefore, minimizing the number of used vias is among the objectives. The quality of the global router solution affects the timing and the power of the chip.

This work was implemented for ICCAD 2019 Contest, LEF/DEF Based Open-Source Global Router. Our implementation starts by parsing the library exchange format (LEF) and design exchange format (DEF), version 5.8. After that, the 3D global routing grid is built. Then, using rectilinear steiner tree, the nets are ordered in descending order of wirelength. Another implementation of steiner tree is then used to decompose the multi-pin nets to two-pin subnets. Following this, A star (A*) is used as the routing algorithm on the two-pin subnets, but with a modified cost function. Finally, the output guide file is printed in ISPD-2018/2019 format.

The remaining of the paper gives an overview of global routing approaches (Section II). Section III outlines OGRE implementation and its core algorithms. In Section IV we show results obtained by running OGRE against benchmark designs. Finally, we conclude the paper in Section V.

II. Approaches for Global Routing

Attributed to the criticality of the global routing problem, and the announced global contests, the research and development of new global routing techniques have been encouraged. For instance, NTHU-Route 2.0 improves the solution quality of state-of-the-art router, NTHU-Route. The main objective of this router is to reduce the sum of overflows and reduce the wirelength, and number of vias in multi-layer designs. The first step is projecting the 3D grid onto a plane, then using FLUTE to decompose the multi-pin net to two-pin subnets. Each two-pin subnet is routed with two probabilistic L-shapes. After that, the congested regions are identified, and rip-up and reroute iterations start. The ripped-up two pin subnets are rerouted using monotonic routing. However, if it fails to find an overflow free path, multi-source and multi-sink routing is used. The final step is layer-assignment [4].

BoxRouter 2.0 also works on 2D grids and at the end performs layer assignment. The algorithm uses negotiation-based A* search and topology aware rip-up instead of ripping up the entire net passing through the congested area. Progressive integer linear programming (ILP) is used for the layer assignment to handle blockages, guarantee feasibility and enhance runtime [5].

FastRoute is another global router that works on 2D grids. It uses congestion-driven, via-aware Steiner tree generation to form good starting topologies for multi-pin subnets, then segment shifting is applied to move the routing demand from the congested region without increasing the wirelength. Following this, 3-bend routing technique is used to explore the path between source and sink pins. Also, multi-source multi-sink routing technique is used to connect two subtrees in multi-pin nets. However, virtual capacity is used to guide the maze router away from congested regions, and an adaptive cost function is used to encourage 3-bend routing and maze routing to find less-congested path. Finally, spiral layer assignment is used to extend the 2D solution to 3D [3].

FastRoute 4.0 improves upon FastRoute 2.0 by considering via number optimization throughout the whole global routing process, not just during layer assignment like other global routers. To address this problem, three new techniques were introduced. Firstly, via aware Steiner tree generation is used, utilizing pre-computed net topologies so that the global router will generate less number of vias, while still taking congestion into consideration. Secondly, 3-bend routing is used which does not suffer from long runtime like maze routers, but is effective in via count reduction and congestion reduction. Thirdly, layer assignment with careful ordering takes place. The nets are ordered in increasing order of wirelength / number of pins. As a result, FastRoute 4.0 succeeds to reduce the number of vias by 13.8% compared to FastRoute [13].

As for Multilevel Congestion-Based Global Router, it is hierarchical in nature and it uses a congestion-map that is created before routing to identify the congested edges. The congested edges form the 3-level hierarchy, with the first level having the 2% most congested edges, the next level having the next 2% most congested edges, and the third level having the rest of the edges. The decision then becomes which subnets to route within each level, giving the most congested edges the freedom to find alternate routes. Each level gives rise to an ILP formulation that maximizes the number of routed subnets [6].

III. OGRE Implementation

Figure 1 shows a block diagram of OGRE components which will be discussed in great detail below.

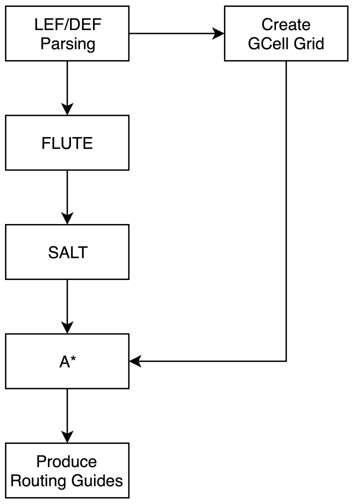

Fig. 1. OGRE Components and Flow

The first step is parsing the LEF and DEF files, For that we use a slightly modified version of SI2 LEFDEF parser [9]. The parsed information is used to construct the Gcell grid. Each Gcell is assigned a maximum routing capacity for wires and for vias separately, given the pitch of each metal layer and the area of the Gcell. Out of the maximum capacity of the Gcell, 75% is given to wires and 25% is given to vias to favor wires over vias.

The parsed nets are, then, ordered descendingly according to their Steiner Tree lengths using FLUTE [8]. Descending ordering is used so that larger nets are routed first. This will result in a minimized detour distance cost if the router finds a blocked path due to another route.

A* is used for path finding (routing). A* is a guided search algorithm and therefore, practically, has a complexity that is far less than other complete search algorithms like Dijkstra's Shortest Path finding algorithm. A* is, also, selected because it has a customizable cost function; this helps tweak the algorithm to perform optimally and enables the algorithm to take many variables into consideration when calculating the cost of going to a certain node. Although we know that A* might not give the most optimal solution for the shortest path between two nodes, the benefit gained in time complexity and cost function customization is a lot more valuable than the minor decrease in optimality of the path finding [2] [11].

We built upon the A* implementation by [10]. However, we changed it from 2D to 3D and introduced to it our cost function. The cost function is designed to achieve our routing objectives:

- Overflow avoidance,
- Congestion reduction, and
- Vias reduction

To avoid overflow, we prevent routing in GCells where the percentage of used resources is greater than or equal to the maximum capacity. In order to reduce congestion and the number of used vias, we implemented a cost function that takes into consideration the used resources of the current GCell.

$$Cost = \alpha \times wire\ utilization + \beta \times via\ utilization \quad (1)$$

α and β are parameters, such that $\alpha + \beta = 1$. Different values of α and β were tried for ICCAD 2019 contest problem C benchmarks. The best tradeoff between wirelength, number of vias, space and short was for α = 0.5, β = 0.5. This does not mean that our implementation does not favor wires over vias, because as aforementioned, the maximum wire capacity is 75% the total Gcell capacity, and the maximum via capacity is 25% the maximum Gcell capacity.

The routing algorithm starts by taking the nets in the descending order out of the maximum priority queue. Then for a given net, the algorithm generates a steiner tree using Steiner shAllow-Light Tree (SALT) [7]. This steiner tree decomposes the multi-pin nets to two-pin subnets. Therefore, it is the algorithm's reference for which points to route to each other using A*. The algorithm then runs a Depth-First Search (DFS) on the Steiner tree to generate a vector that contains the pairs of nodes, in the GCell Grid, to be routed to each other. At this point the algorithm has a vector containing pairs of $(x1, y1), (x2, y2)$ coordinates and begins running A* for each pair to generate the route path between these two points. After all the pairs contained in this net are successfully routed, the algorithm pops out the next net from the maximum priority queue and starts over from generating the SALT Steiner tree and continues till all the nets are successfully processed.

The pseudo-code of our implementation is in algorithm 1. The pseudo-code for metal layer assignment of Steiner tree points is in algorithm 2.

```
Parse the design LEF and DEF files;
Construct Gcell grid;
Assign wire capacities and via capacities to GCells;
Sort, descending order, the nets using FLUTE;
Place sorted nets into a queue (Q);
while Q is not empty do
    Use SALT to decompose the net into two-pin subnets;
    Assign metal layers to Steiner points by calling
      ASSIGN_LAYERS_DFS(child,
      current_node_parent);
    foreach subnet do
        Route using A*;
        Produce routing guide;
    end
end
```

Algorithm 1: OGRE Algorithm

```
procedure ASSIGN_LAYERS_DFS(child,
current_node_parent) ;
if the current node is a Steiner node then
    Current_node_parent = parent layer;
end
else
    Current_node_parent = layer of current node from
    the def or lef file;
end
foreach child of the current node do
    Push the parent node and child node in the vector of
    nodes to be routed (2-pin subnets);
    ASSIGN_LAYERS_DFS(child, current_node_parent);
end
```

Algorithm 2: Layer Assignment Algorithm

IV. Experimental Results

To evaluate OGRE, it was compared to the OpenRoad project global router which implements FastRoute 4.0 [13] algorithm. We ran both against the ICCAD 2019 global routing contest benchmark designs. The resulted routing guides are then fed to Dr. CU [12] detailed router. The tests were run on ubuntu 64 bits distribution in an oracle virtual machine with 14 cores and a memory of 28GB, with nested paging on. Table I shows the obtained results after detailed routing is successfully run on the designs. Here we focus on the total wirelength, total number of vias, the number of minimum spacing DRC violations and short DRC violations as reported by Dr. CU.
Ogre showed a performance very close to FastRoute 4.0. In some cases it outperformed FastRout 4.0.

As it can be seen from Table I comparing between OGRE and FastRoute 4.0, in the 5 recorded benchmarks, OGRE always had less spacing violations than FastRoute 4.0. In 2 benchmarks, OGRE had less short violations. The high number of short violations obtained by OGRE in ISPD18-3 and ISPD19-3 is attributed to currently not handling obstructions. In other words, the capacity of the GCells is not reduced when part of it, or its entirety is blocked. As for the wirelength, OGRE had the lower wirelength in the 2 highest wirelength designs of the 5 benchmarks.

TABLE I
OGRE vs. OpenRoad FastRoute 4.0

	OGRE			
	Wirelength	*no. of vias*	*short vios*	*spacing vios*
ISPD18-1	475770	36563	0.3	0
ISPD18-2	7287680	396200	140.83	32
ISPD18-3	7533960	374115	14428.1	41
ISPD19-1	693784	39363	1186.5	147
ISPD19-3	910578	67347	6776.88	222

	FastRoute 4.0			
	Wirelength	*no. of vias*	*short vios*	*spacing vios*
ISPD18-1	466586	31976	5.41	2
ISPD18-2	8276470	346712	427.14	189
ISPD18-3	9181020	350013	630.15	235
ISPD19-1	684564	37599	1109	193
ISPD19-3	896482	59771	150.32	373

V. Conclusion

In this paper, we develop a new open-source global router, OGRE, that focuses on reducing the congestion, the number of vias and total wirelength. When compared to FastRoute 4.0 (one of the best, openly available, academic global routers), it produced close or better results.
For future work, we plan to handle obstructions by reducing the capacity of the obstructed GCells. Moreover, we plan to use machine learning techniques to predict the best global routing parameters given a design. Finally, we plan to implement hierarchical congestion to balance the congestion of the whole GCell grid and implement rip and reroute for the nets in the congested regions.

VI. Acknowledgment

OGRE is a publicly available, open-source project that can be found at https://github.com/Cloud-V/OGRE.
The project was tested on Apple macOS Mojave Version 10.14.6 and Ubuntu 18.04.3 LTS.

References

[1] Zhang, Yanheng, "Handling the complexity of routing problem in modern VLSI design," graduate theses and dissertations, Iowa State University, 2011. DOI: https://doi.org/10.31274/etd-180810-2348
[2] Chen, Huang-Yu , "Global and Detailed Routing", Electronic Design Automation: Synthesis, Verification, and Test (Systems on Silicon) Chapter 12 Mar. 12, 2009 , pp. 687-750.
[3] M. Pan and C. Chu, "FastRoute 2.0: A High-quality and Efficient Global Router," 2007 *Asia and South Pacific Design Automation Conference*, Yokohama, 2007, pp. 250-255. doi: 10.1109/ASPDAC.2007.357994
[4] Y. Chang, Y. Lee and T. Wang, "NTHU-Route 2.0: A fast and stable global router," 2008 *IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, 2008, pp. 338-343. DOI: 10.1109/ICCAD.2008.4681595

[5] Minsik Cho, Katrina Lu, Kun Yuan and D. Z. Pan, ”BoxRouter 2.0: architecture and implementation of a hybrid and robust global router,” 2007 *IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, 2007, pp. 503-508. DOI: 10.1109/ICCAD.2007.4397314

[6] Logan Rakai, Laleh Behjat, Shawki Areibi, and Tamas Terlaky, A Multilevel Congestion-Based Global Router, VLSI Design, vol. 2009, Article ID 537341, 13 pages, 2009. DOI: https://doi.org/10.1155/2009/537341.

[7] G. Chen, P. Tu and E. F. Y. Young, ”SALT: Provably good routing topology by a novel steiner shallow-light tree algorithm,” in 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Irvine, CA, 2017, pp. 569-576. doi: 10.1109/ICCAD.2017.8203828

[8] C. Chu, ”FLUTE: fast lookup table based wirelength estimation technique,” in IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004., San Jose, CA, USA, 2004, pp. 696-701. doi: 10.1109/ICCAD.2004.1382665

[9] LEFDEF Utilities, https://github.com/jinwookjungs/lefdef_util

[10] GH repo: https://github.com/justinhj/astar-algorithm-cpp

[11] G.T. Heineman G. Pollice and S. Selkow. Algorithms in a Nutshell O’Reilly 2009.

[12] G. Chen, C. Pui, H. Li and E. F. Y. Young, ”Dr. CU: Detailed Routing by Sparse Grid Graph and Minimum-Area-Captured Path Search,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[13] Yue Xu, Yanheng Zhang and Chris Chu, ”FastRoute 4.0: Global router with efficient via minimization,” 2009 Asia and South Pacific Design Automation Conference, Yokohama, 2009, pp. 576-581.