EE215A: Router

An Lihua
2022131001

Gong Yutao
2022231081

***Abstract*—In this project, we have built a real maze-router and run on some synthetic benchmarks and on some real industrial benchmarks. We have achieved 100% completion in all benchmarks and optimized to minimize pathcost as much as possible. The optimization results are very significant in industrial benchmarks.**

***Index Terms*—Router, Dijkstra**

I. Introduction

Routing is the process of determining the interconnect paths for the components and circuit elements on an integrated circuit (IC) design. Routing plays a crucial role in the physical design phase of IC fabrication.

When designing complex ICs with millions or even billions of transistors and interconnections, it becomes essential to efficiently route the interconnections to achieve optimal performance and meet design specifications. It requires sophisticated algorithms, optimization techniques, and computational tools to efficiently and accurately route the interconnections while meeting various design constraints, such as timing, power, and manufacturability.

Here are some basic problems of current routing method. First is the scale, big chips always have an enormous number (millions) of wires, not every wire gets to take an "easy" path to connect its pins. We must connect all of them, but we can not afford to route many wires manually. Second is the geometric complexity. At nanoscale, geometry rules are complex, which makes routing hard. Nevertheless, we can use a simple grid representation of layout to start. Finally, is the electrical complexity. It's not enough to make sure you connect all the wires. Also need to ensure that the delays through the wires are not too big and that wire-to-wire interactions do not mess up behavior.

There are not quite so many routing algorithms but lots of routing data structures to represent the geometry efficiently. But there is one very big idea at the core of most real routers, which is called "Maze Routing".

In this project, we have built a real router based on this idea, then optimized in algorithm, data structure, input processing, routing priority, and other aspects. Finally, we compared the results before and after optimization, which gives a significant improvement.

II. Basic Algorithm

Our basic algorithm follows the ideal of "Maze Routing". The basic strategy is very simple: completely wire one net at a time, then move on next net to find the best path. We do this by three steps: (1) Expand; (2) Backtrace; (3) Clean up.

From Fig. 1 (a), we expand one cell at a time until a shortest path from S to T are found. Expansion creates a wavefront of paths that search broadly out from source cell until target is reached. "Reached" means specifically we label it with a pathlength number, a queue of reachable squares from the start pin is used. The wavefront is expanded under a specific order we defined, and expanded the minimum cost direction first.

Fig. 1 (b) shows the process of backtrace. We first select a shortest path (any shortest path) from target back to source and mark its cells so they can not be used again. Finally, we clean up the grid for the next net as shown in Fig. 1 (c).

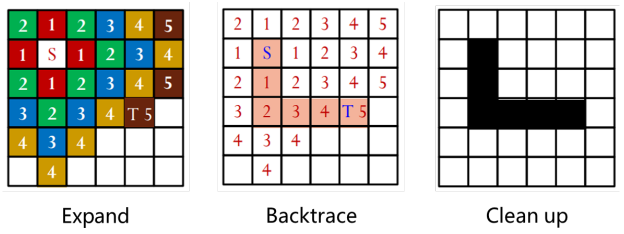

Fig. 1. Diagram of maze router.

However, this basic, simple algorithm has some problems: (1) early nets wired may block the path of later nets, which will lower our completion rate; (2) we have to search the whole grid each time, which will take a long time to finish routing; (3) this method does not give us guaranteed cheapest path due to the expensive penalty of bends and vias.

We have shown the results of completion rate and the pathcost in Table I. and we can clearly see that the completion rate for industrial benchmarks has significantly decreased, while the cost for simple benchmarks is very high. Therefore, we have made optimizations to address some of the issues mentioned above.

TABLE I
Results of Maze Router.

bench	Accuracy	Cost
bench1	20/20	352
bench2	20/20	3110
bench3	16/16	1279
bench4	15/15	1793
bench5	86/128	9433
fract2	109/125	12142
primary1	679/830	169094
industry1	839/1000	572584

III. Optimization

A. Algorithm and Data Structure

For the algorithm level, we use the Dijkstra's algorithm, which works for gridded maze routing. We assume wavefront is a cost-indexed list of cells already visited during search, and "labeled" with pathcost. We also introduce the concept of predecessor to record the direction of each reached cell. At the same time, we take into account the bends and vias penalty when expanding the wavefront, which guarantee that we can find the cheapest path. For the data structure, we choose the min heap structure, which is a complete binary tree.

B. Net Order

We have adjusted the order of the inputs. We calculated the absolute distances of all nets and sorted them by distance. We wired the shorter nets first then the longer nets. The advantage is shown in Fig. 2. If we follow the original order, there may be a long net separating the grid, causing other nets to be unable to connect. However, if we connect the shorter nets first, enough space can be reserved for the longer nets.

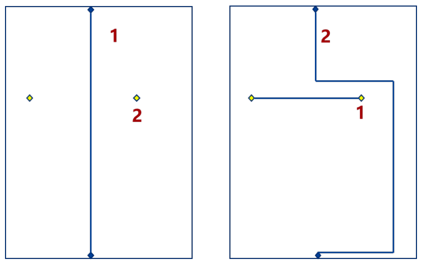

Fig. 2. Net order.

C. Preferred Routing Directions

The way a real, industrial-strength router manages to route millions of nets on 8-12 layers of physical metal routing, is to impose preferred direction constraints on the layers. This is most commonly done by restricting alternating layers to be "mostly vertical" and "mostly horizontal". In this project, we make Layer1 strictly-vertical routing, and Layer2 strictly horizontal. This means we will use a lot of vias. But this also means we will not block wires on one layer and increase our completion rate. The idea is illustrated in the Fig. 3.

We have also considered the preferred one-way routing, which allow a "wrongway" penalty. However, after our experiments, this method did not bring good results, but even worse results for some benchmarks, so we abandoned this plan.

IV. Results

In order to display our results more conveniently, we conducted visualization processing, displaying the shape of the grid and the routing results.

Bench1 (Fig. 4) is one layer unit-cost, there are no bend penalties and all the routing are on just one layer. Since it is the most simple benchmark, the result before and after optimization is not obvious.

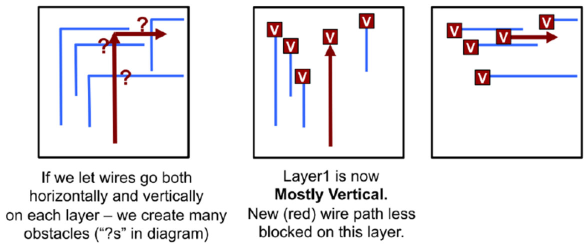
?
?
?
If we let wires go both horizontally and vertically on each layer – we create many obstacles ("?s" in diagram)
V
Layer1 is now **Mostly Vertical.** New (red) wire path less blocked on this layer.

Fig. 3. Preferred routing directions.

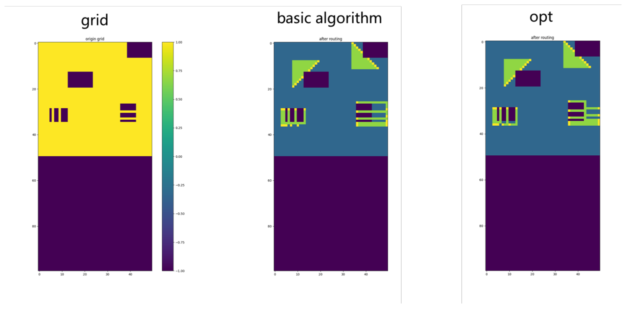

Fig. 4. bench1

Bench2 (Fig. 5) is one layer with bend penalty and non-unit cost. In this bench, there is an obvious difference in the results. Before optimization, our router will ignore high penalty areas and choose the shortest path, but after optimization, our router will choose the path with the lowest cost.

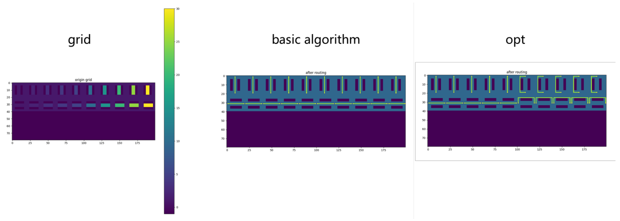

Fig. 5. bench2

Bench3 (Fig. 6) is a no-vias 2-layer router. There are nets that can be routed entirely on Layer1, and some nets that can be routed entirely on Layer2. The results are similar with bench2.

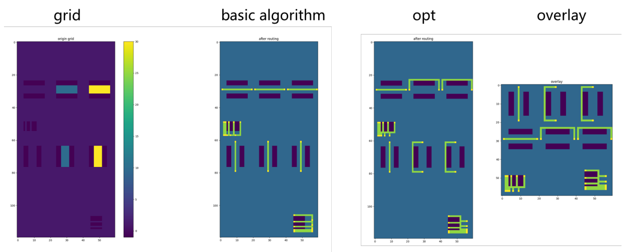

Fig. 6. bench3

Bench4 (Fig. 7) is real 2-layer router. This means we have to deal with vias in this bench.

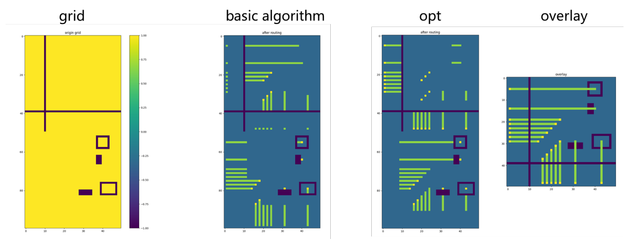

Fig. 7. bench4

So far, we have completed the basic benchmarks, next are large-scale industrial routing (Fig. 8-11). We can see that the pre-optimized routing is chaotic and cannot achieve a 100% completion rate. However, for the optimized routingresults, we can clearly see that Layer 1 is vertical routing, and Layer 2 is horizontal routing, while greatly reducing cost.

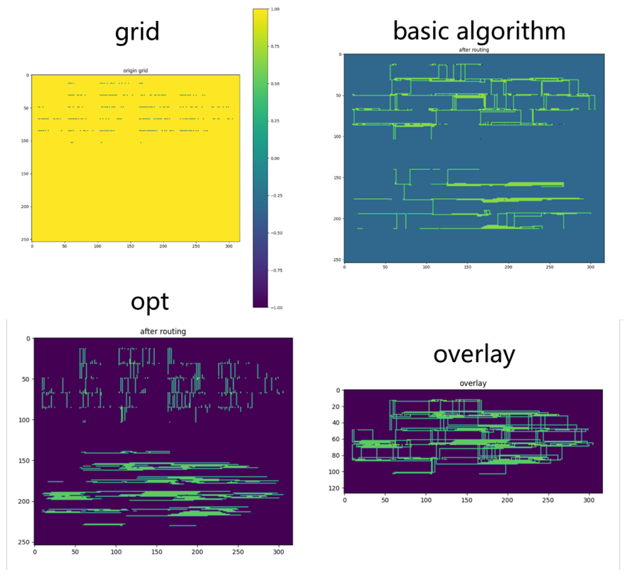

Fig. 8. bench5

Finally, we summarized all the results in Table II. By comparison, it can be seen that after optimization, we achieved a 100% completion rate for all benchmarks, while greatly reducing costs and achieving good results.

V. Conclusion

In this project, we have built a real maze-router and run on some synthetic benchmarks and on some real industrial benchmarks. We have optimized in algorithm, data structure, input processing, routing priority, and other aspects. Finally, we compared the results before and after optimization, achieved 100% completion in all benchmarks and optimized to minimize pathcost as much as possible. The optimization results are very significant in industrial benchmarks.

References

[1] Hao Geng. EE215A: EDA, 2022-2023 Spring

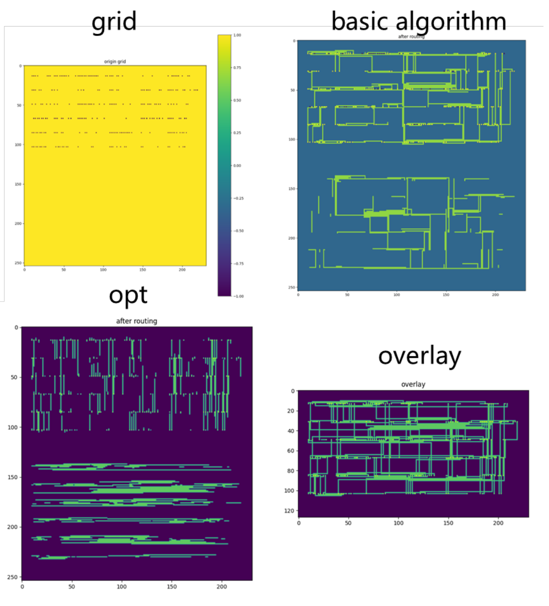

Fig. 9. fract2

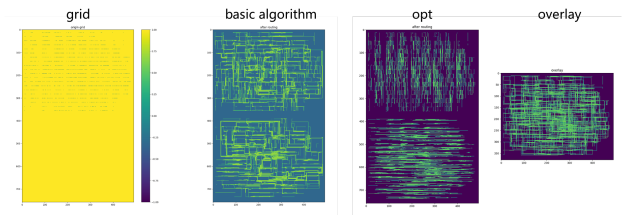

Fig. 10. primary1

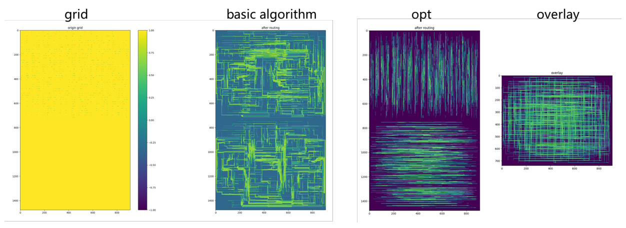

Fig. 11. industry1

TABLE II
Results after Optimization.

bench	Accuracy	Accuracy after Opt	Cost	Cost after Opt
bench1	20/20	**20/20**	352	**372**
bench2	20/20	**20/20**	3110	**1760**
bench3	16/16	**16/16**	1279	**479**
bench4	15/15	**15/15**	1793	**1793**
bench5	86/128	**128/128**	9433	**12070**
fract2	109/125	**125/125**	12142	**11892**
primary1	679/830	**830/830**	169094	**119763**
industry1	839/1000	**1000/1000**	572584	**406117**