ASIA SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE

A High Performance Detailed Router Based on Integer Programming with Adaptive Route Guides

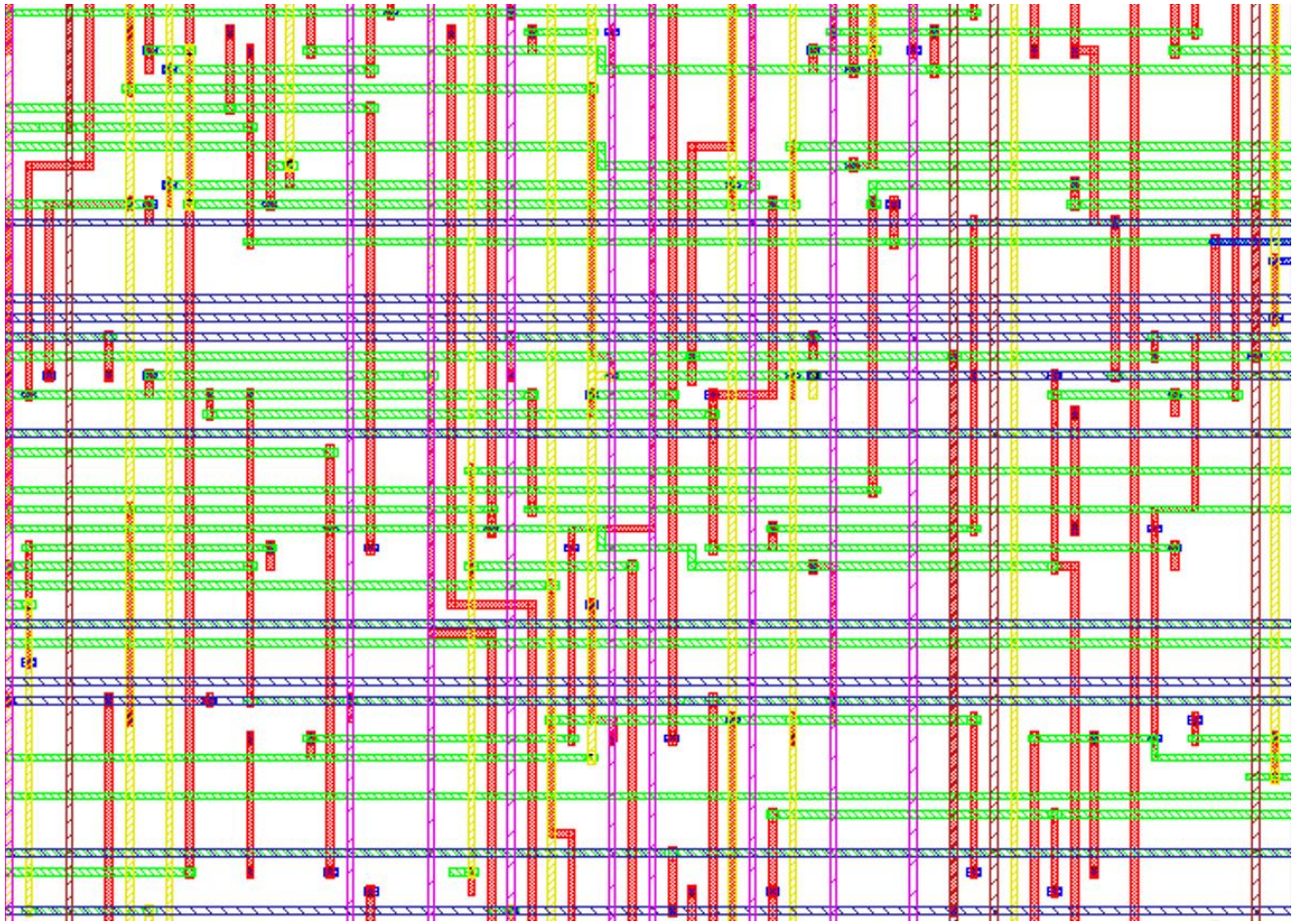
Zhongdong Qi, Shizhe Hu, Qi Peng, Hailong You, Chao Han, Zhangming Zhu

School of Microelectronics, Xidian University

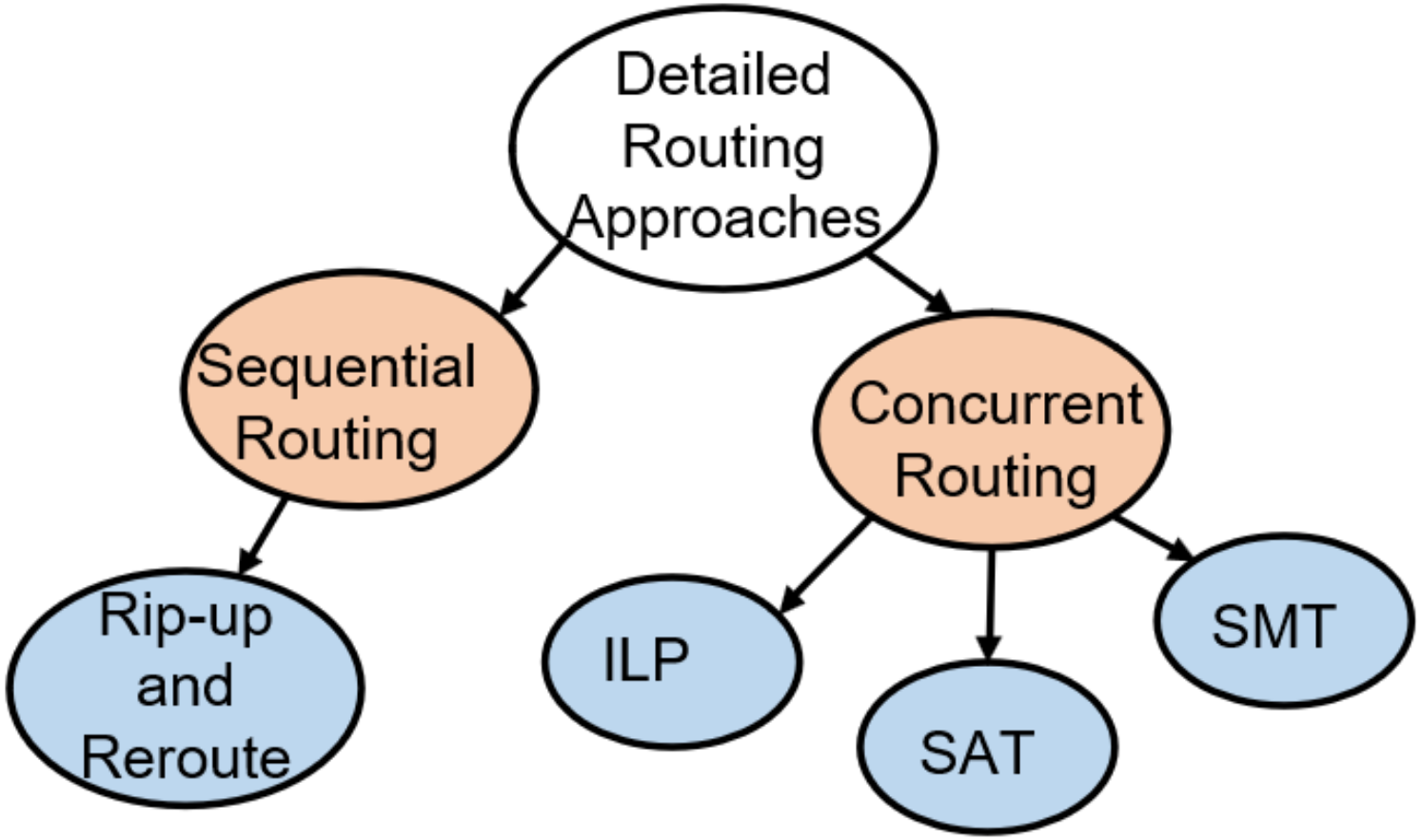
OUTLINE

01 Introduction

➢ Detailed Routing

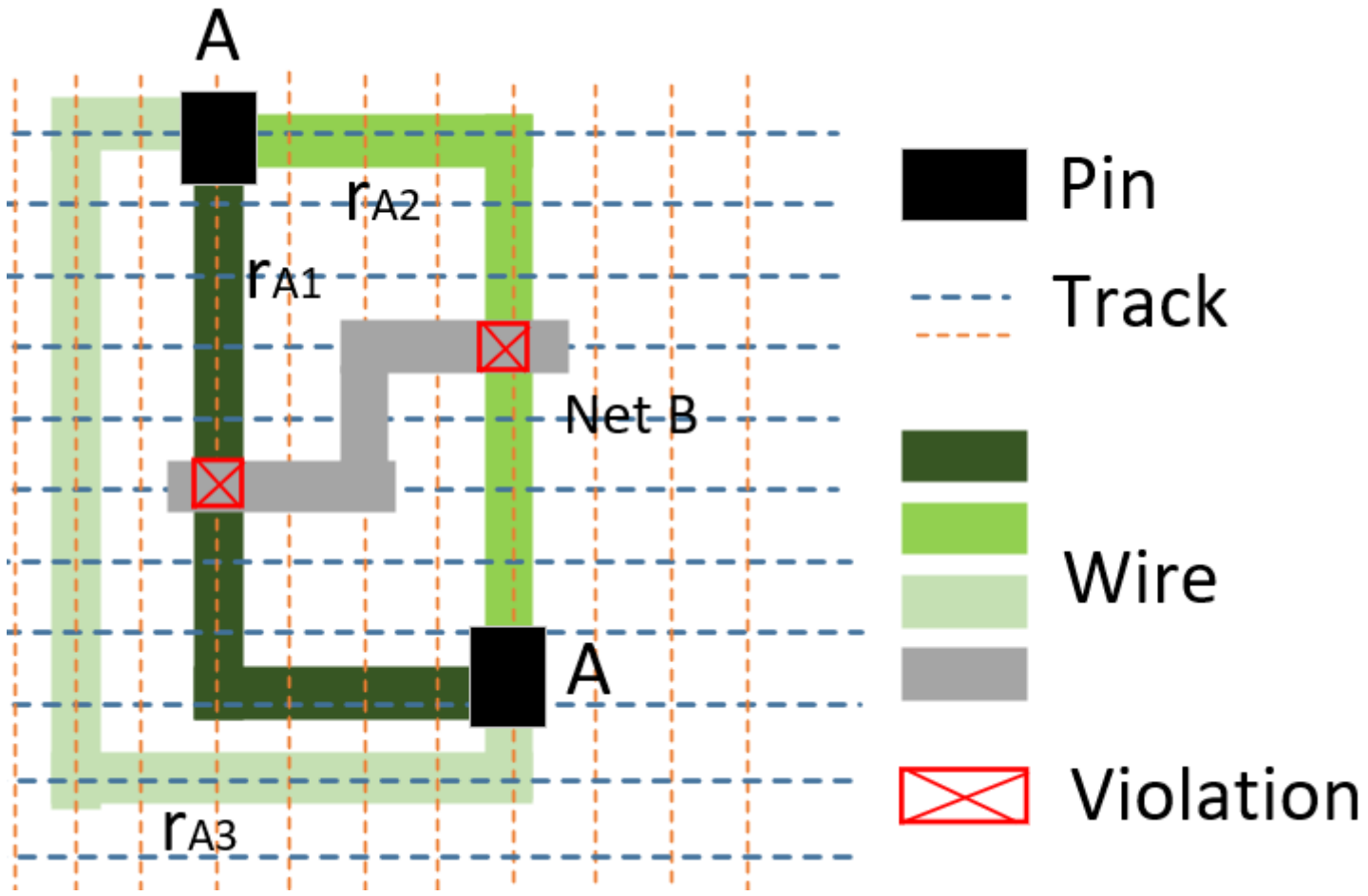


➢ Introduction



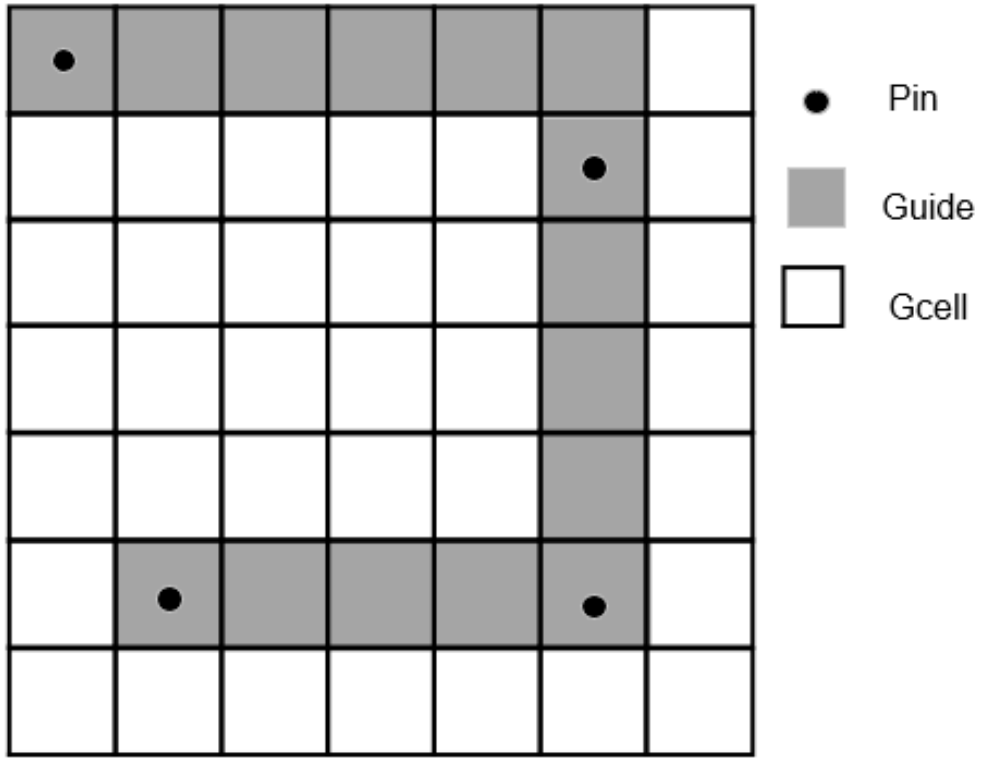
Detailed Routing Approaches

➢ Motivation and Goal



- Net routes ripped-up are discarded too early in sequential routing.
- The ordering of nets has a huge impact on the quality of the solution.

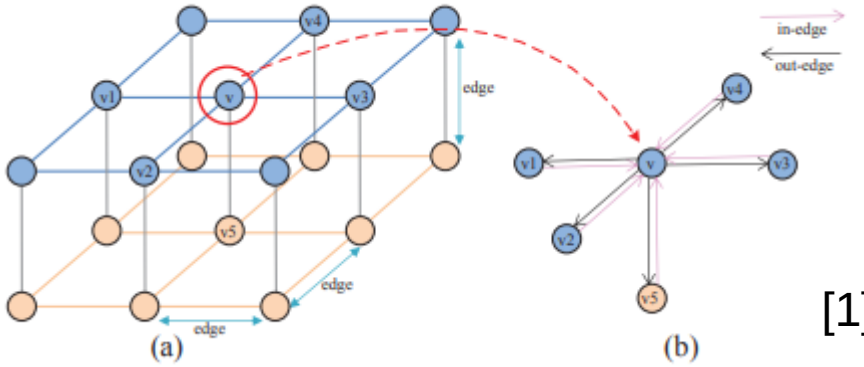
➢ Motivation and Goal



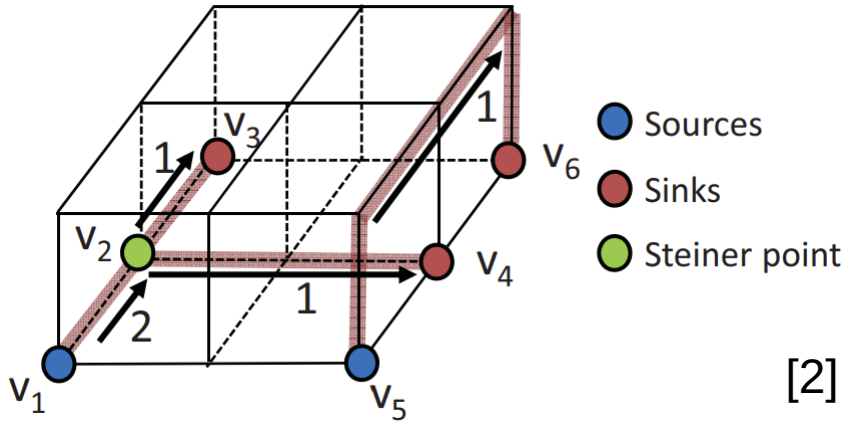
- The quality of the guide can impact the result of detailed routing.
- Detailed routing paths are searched with less flexibility by strictly following guides.
- Guide has a global view of path finding, but lacks of detailed aspects.

➢ Motivation and Goal

The concurrent routing approach



[1]



[2]

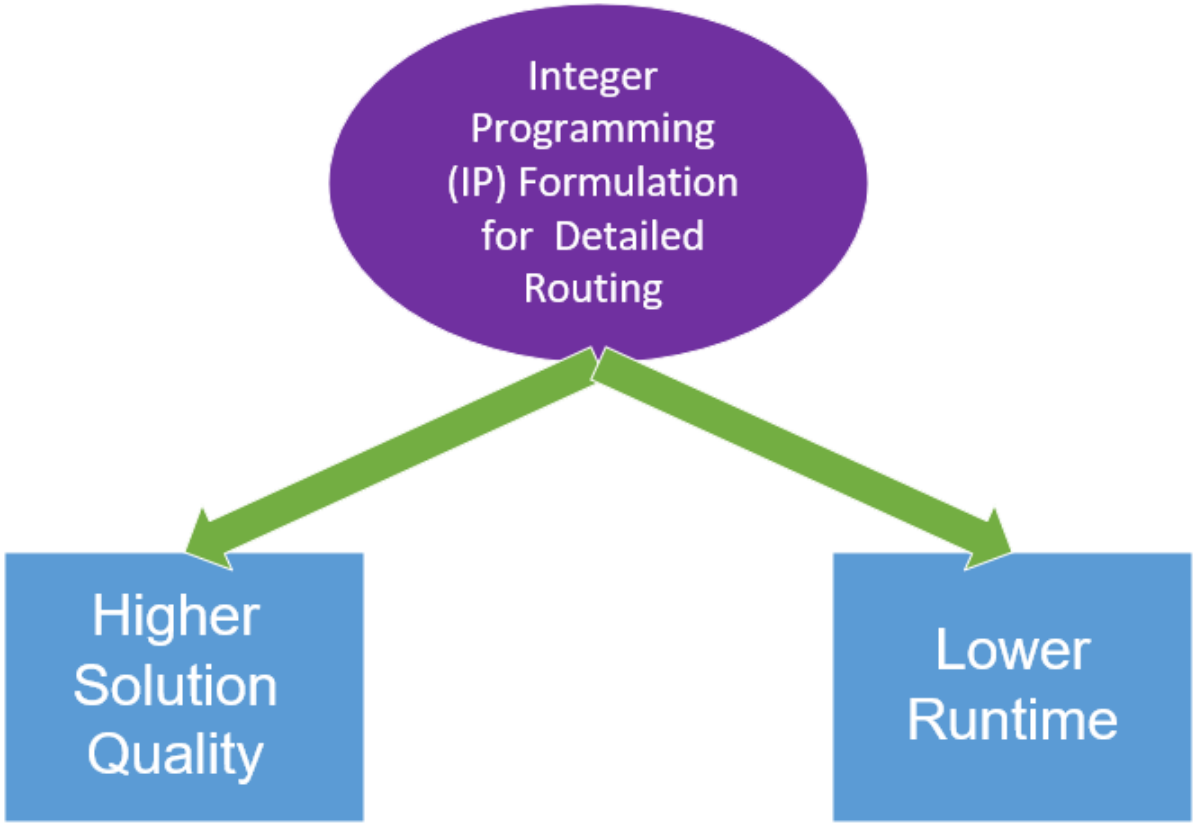
- A large number of grids need to be solved.
- Complex design rules are difficult to consider.
- Path which is not on grids is not easy to be generated.
- It generates high quality solutions, but requires long runtime.

[1] X. Jia, Y. Cai, Q. Zhou, and B. Yu. 2018. “A Multicommodity FlowBased Detailed Router With Efficient Acceleration Techniques.” TCAD 37, 1 (2018), 217–230

[2] K. Han, A. B. Kahng, and H. Lee. 2015. “Evaluation of BEOL Design Rule Impacts Using An Optimal ILP- based Detailed Router.” DAC, 68:1–6.

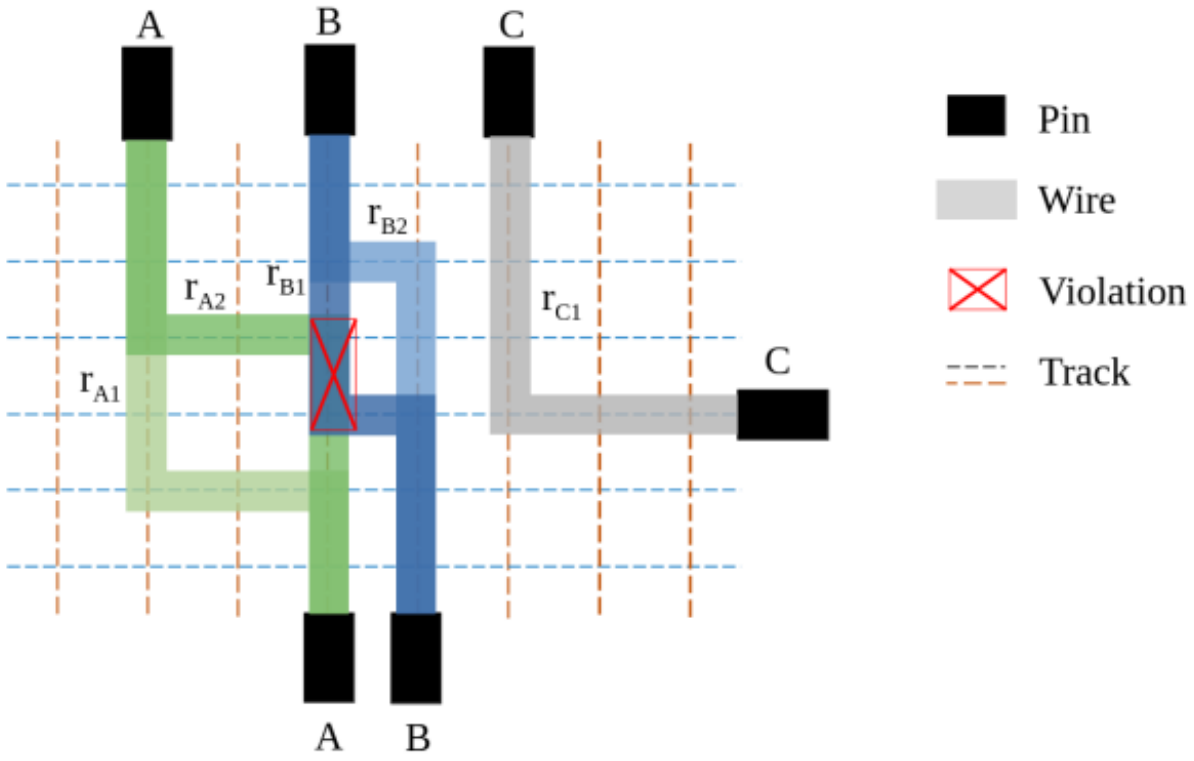
➢ Motivation and Goal

Goal: To achieve higher quality routing results and reduce runtime in detailed routing.



02 | Algorithms

➢ Problem Instance

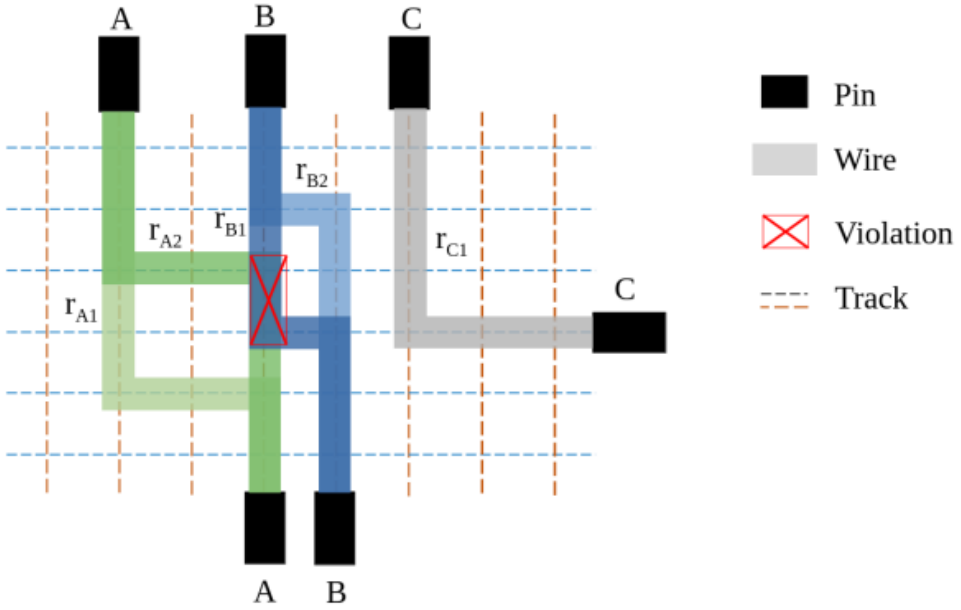


- To meet minimum cost , rA2 and rB1 would not be selected simultaneously for net A and net B.

➢ Integer Programming Formulation

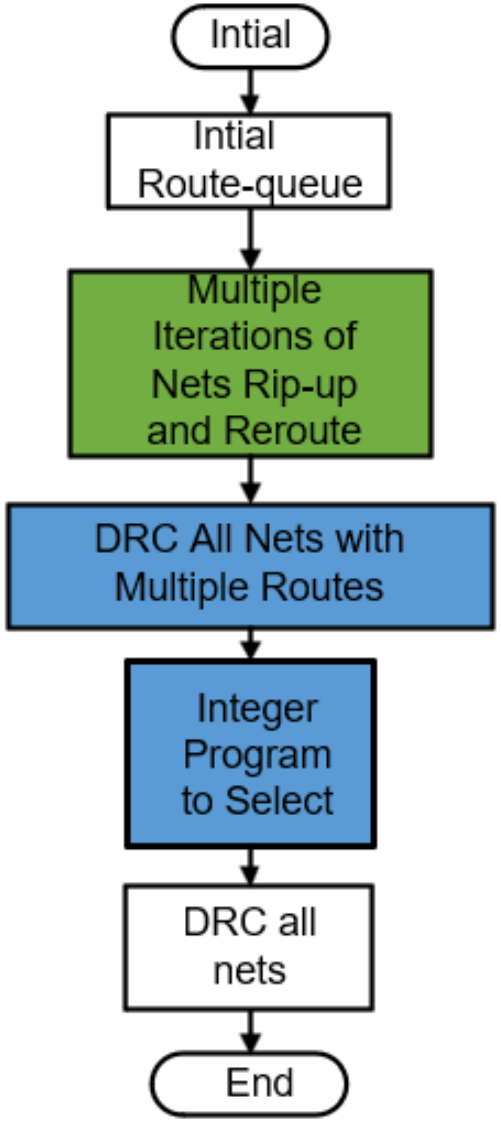
$$\min_{x} \sum_{i=1}^{N} \sum_{t \in \mathcal{T}_i} c_{it} x_{it} + \sum_{i=1}^{N} \sum_{u \in \mathcal{T}_i} \sum_{j=i+1}^{N} \sum_{v \in \mathcal{T}_j} p_{iu,jv} x_{iu} x_{jv}$$

$$s.t. \quad \begin{cases} \sum_{k \in \mathcal{T}_i} x_{ik} = 1 & \forall i = 1, 2, ..., N \\ x_{ik} = \{0, 1\} & \forall i = 1, 2, ..., N, \forall k \in \mathcal{T}_i \end{cases}$$



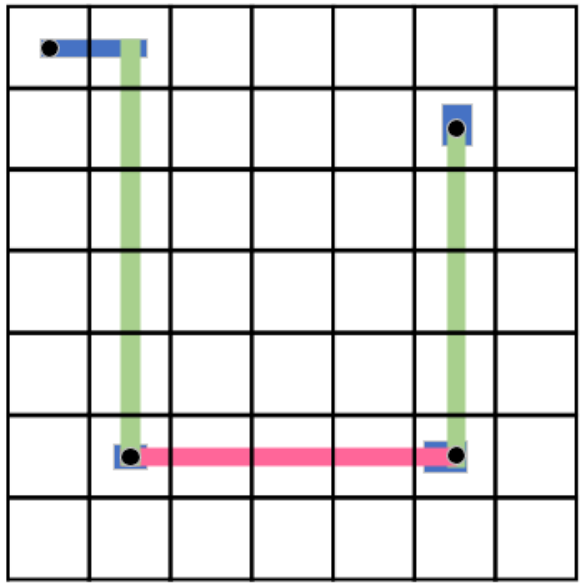
- x_{ik} is an integer variable that takes the values 0 or 1
- $x_{ik} = 1$ means the route k of net n_i is selected .

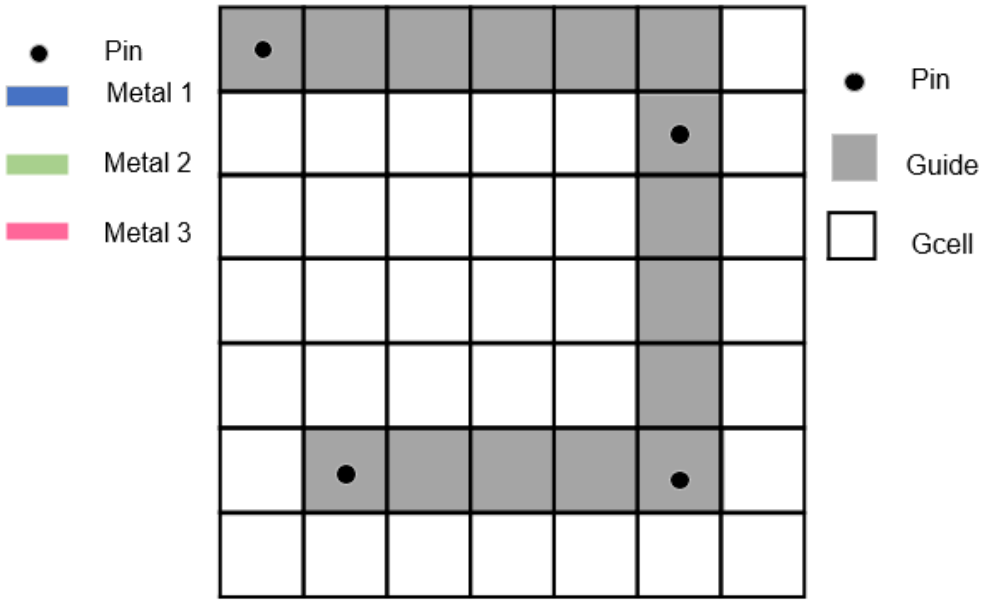
➢ Process in Local Routing Region



Multiple iterations of nets ripup and reroute step :
1. Generate high quality route with adaptive guide following.
2. Collect the paths generated during this step for integer programming sloving.

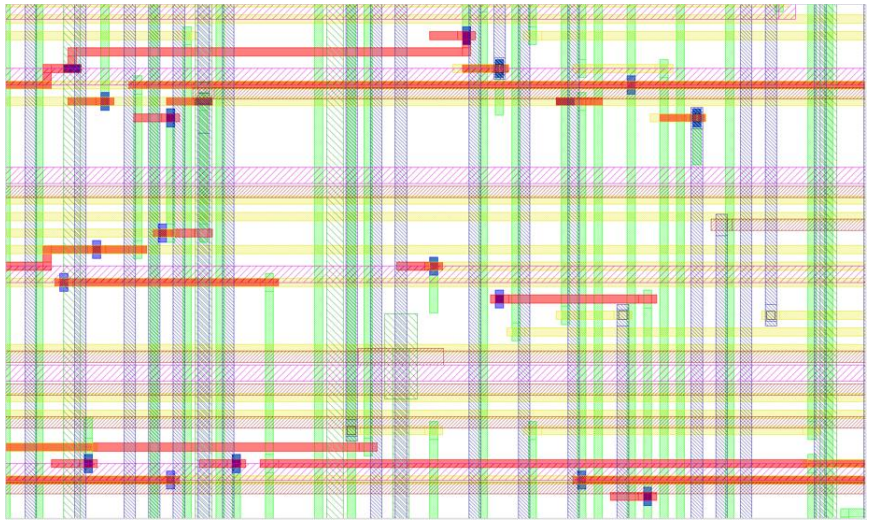
➢ Generate Route with Adaptive Guide



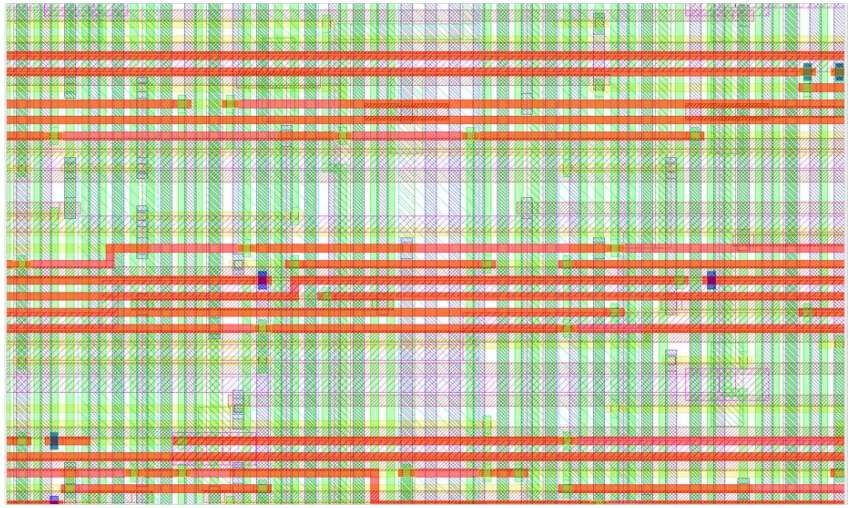


Detailed routing unsatisfy guide

Guide



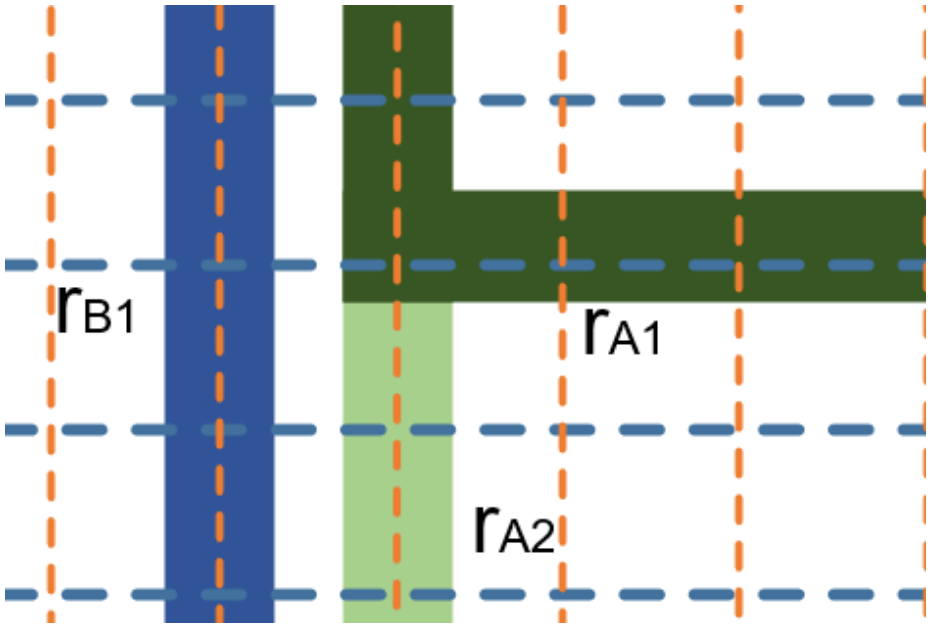
test9

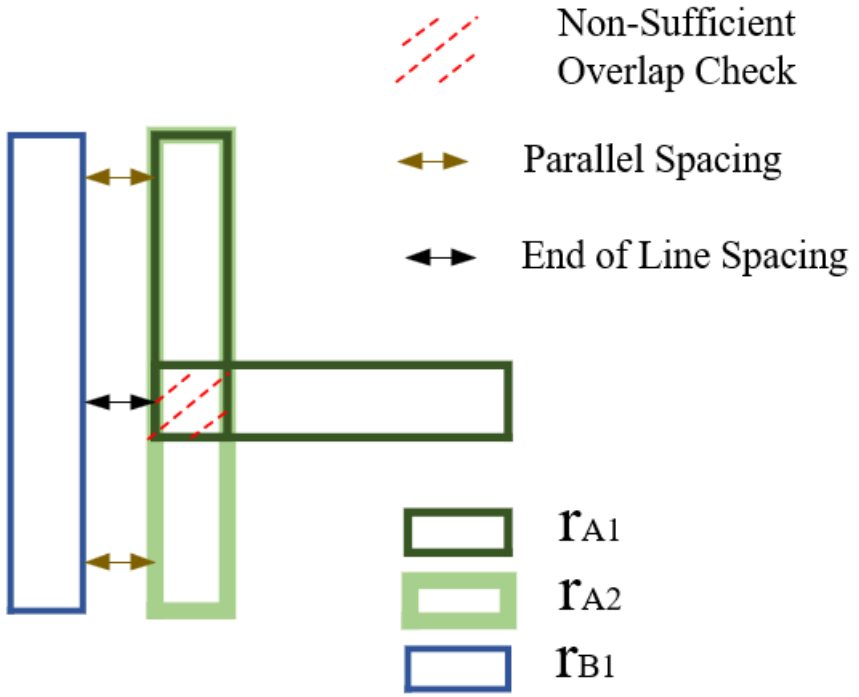


test10

- To generate high quality route, we set router no need to follow guide, when design is low congestion.
- Following guide is necessary when design is high congestion.

➢ Multi-Route Design Rule Checking





- Violations are generated by comparing either different nets or the same net, excluding comparisons of different routes within the same net.
- When check violations related to r_{A1}, rectangles belong to r_{A2} while be ignored.
- Parallel Spacing Check: $r_{A1} \leftrightarrow r_{B1}$ $r_{A2} \leftrightarrow r_{B1}$

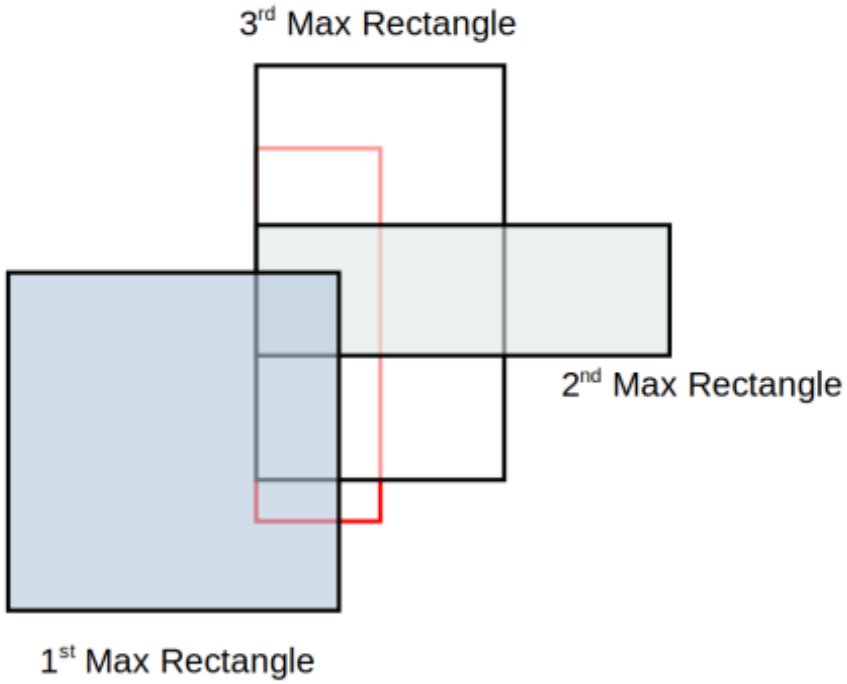
  End of Line Spacing Check: $r_{A1} \leftrightarrow r_{B1}$

  Non-Sufficient Overlap Check: $r_{A1} \leftrightarrow r_{A1}$

➢ Multi-Route Design Rule Checking

Algorithm 3: Check Non-Sufficient Metal Overlap

```
Input : Max rectangle m, n
1 nsRect ← getIntersection(m, n);
2 if diagLen(nsRect ≥ minWidth) then
3     return;
4 if width(m) < minWidth or width(n) < minWidth then
5     return;
6 if hasValid3rdObjWithRoute(nsRect) then
7     return;
8 addMarkerWithRouteInfo(NSMetOverlap);
```



- The step in the red border is designed to filter out different paths within the same net.
- Comparing with others, the rectangle with red outline belongs to the same net but different routes, then it needs to be filtered out.
- Ours engine can check nets with multiple routes simultaneously.

03 Experimental Results

➢ Experimental Environment

Circuit	#StdCell	#Block	#Net	#Layer	Tech Node
test1	8879	0	3153	9	45nm
test2	35913	0	36834	9	45nm
test3	35973	4	36700	9	45nm
test4	72090	4	72410	9	32nm
test5	71946	8	72394	9	32nm
test6	107919	0	107701	9	32nm
test7	179865	16	179863	9	32nm
test8	191987	16	179863	9	32nm
test9	192911	0	178858	9	32nm
test10	290386	0	182000	9	32nm

Information of ISPD-2018 benchmark circuits

- Detailed routing algorithm in C++ programming language
- Use CPLEX as the IP solver
- Use ISPD 2018 detailed routing benchmark suite
- Use TritonRoute-WXL to perform route searching
- Modified design rules are based on TritonRoute-WXL DRC engine

➢ Experimental Results

Circuit	Wirelength (μm)		Via Count		#DRV		Runtime (sec)	
	TR	Ours	TR	Ours	TR	Ours	TR	Ours
test1	86440	85851	35416	35459	0	0	60	54
test2	1572819	1570490	361079	355329	0	0	722	662
test3	1751892	1750143	360515	356714	0	0	1132	946
test4	2621650	2617382	729868	725305	7	2	2898	2186
test5	2763865	2759353	906166	821316	0	0	1773	1680
test6	3551832	3546875	1369534	1227756	0	0	2847	2534
test7	6475063	6466423	2228509	2017927	0	0	5524	4930
test8	6503730	6494203	2245372	2038621	0	0	4988	5004
test9	5433663	5425144	2238814	2025410	0	0	4254	4288
test10	6759905	6759507	2419202	2418613	0	0	5722	6474
Average	1.000	0.998	1.000	0.949	–	–	1.000	0.928

Results between TritonRoute-WXL detailed router (TR) and IPAG (Ours).

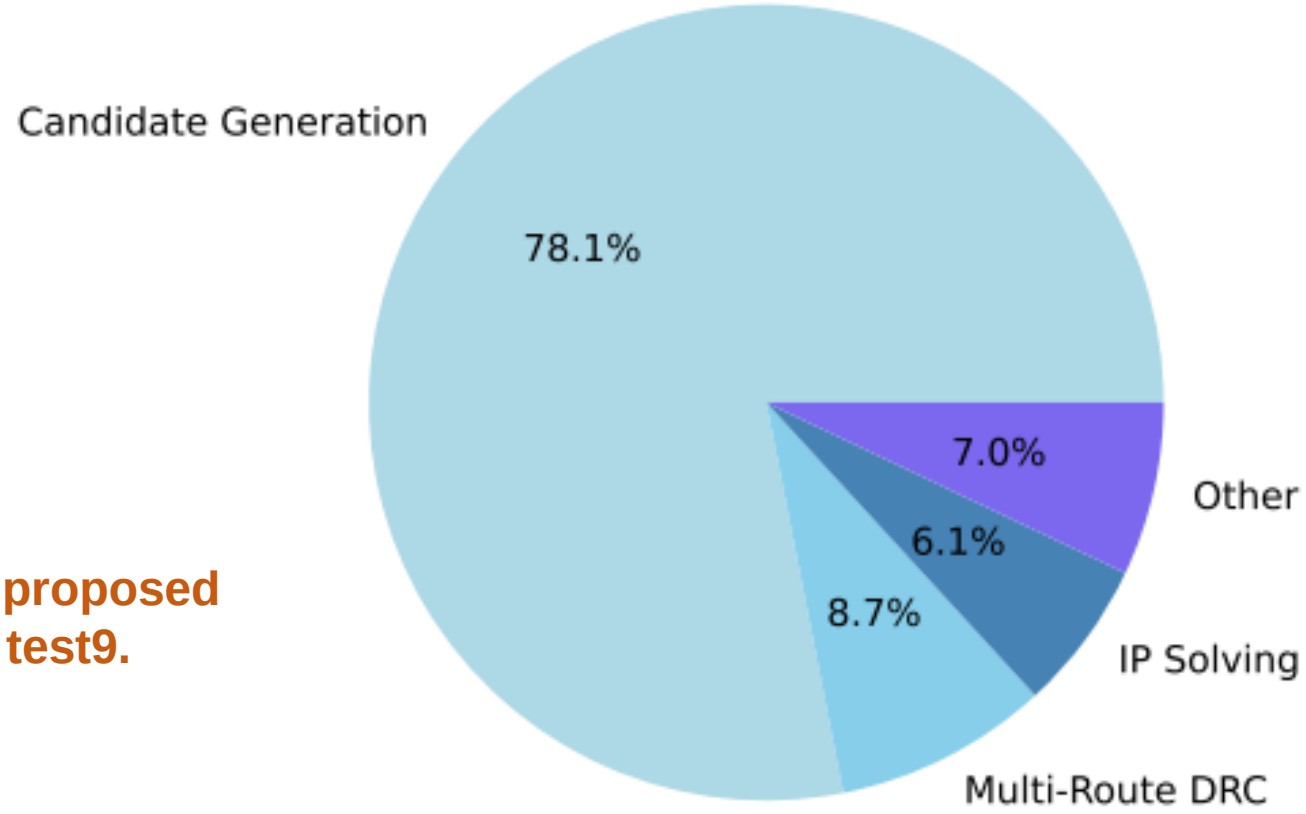
➢ Experimental Results

The number of design rule violations after certain number of DR iterations (#Iter) on test7.

#Iter	0	1	2	3	4	5	8	9	16	17
TR	42762	1112	221	10	6	5	2	2	1	**0**
Ours	41675	573	63	7	5	4	1	**0**	n/a	n/a

- The number of DRV by ours is smaller than TR after each iteration
- Through IPGA selecting candidate routes: fewer rip-up and reroute
  fewer iterations
  fewer runtime

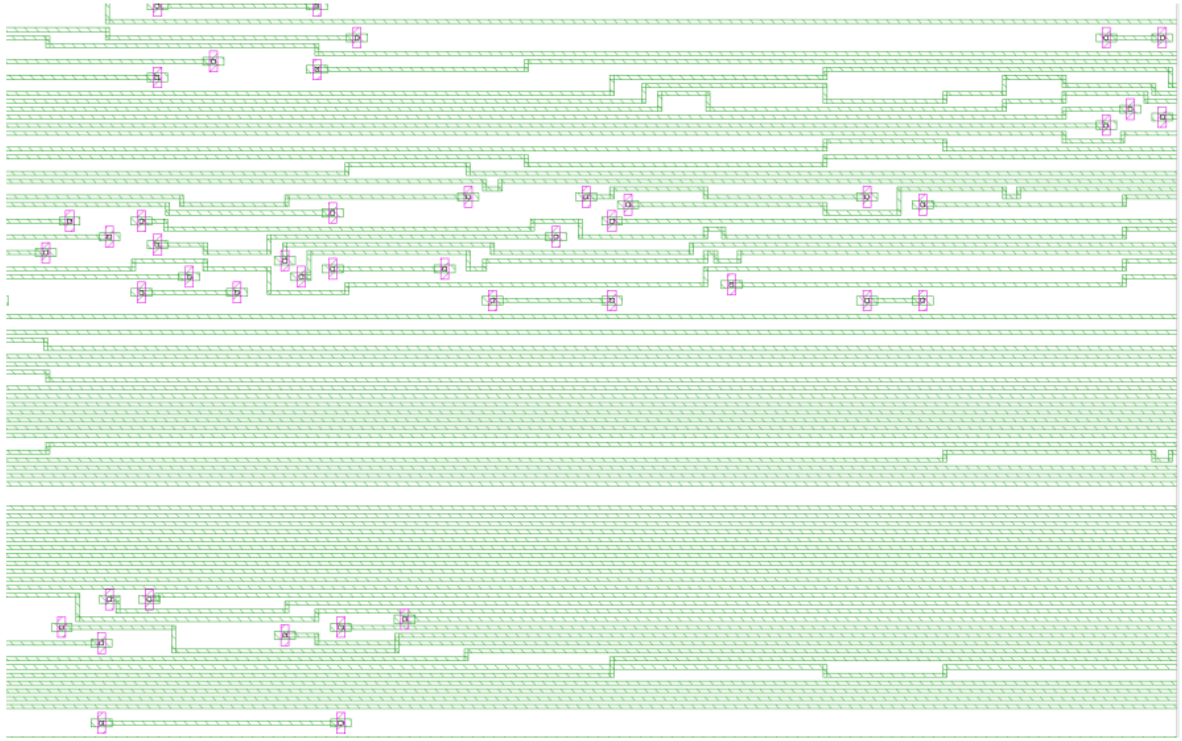
➢ Experimental Results



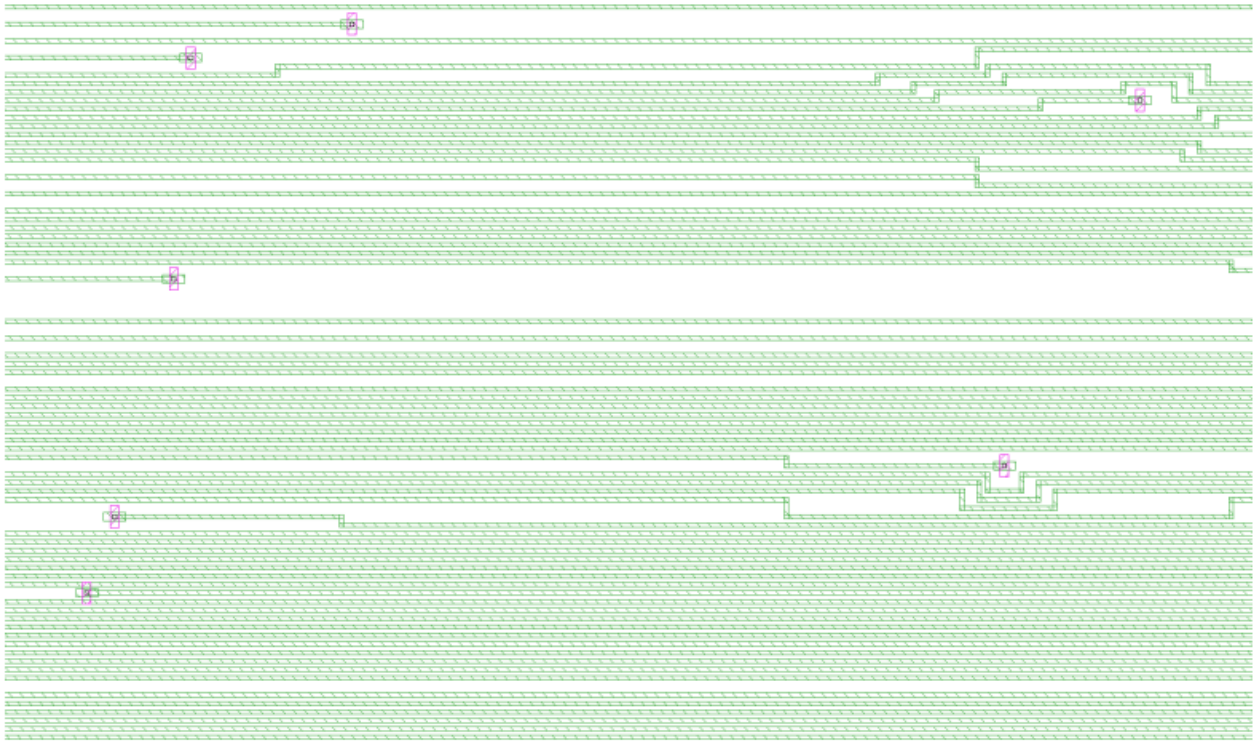
Runtime breakdown of proposed detailed routing on test9.

- Solving IP instances takes only a fraction of runtime.
- Multi-route design rule checking takes similar runtime as IP solving .

➢ Experimental Results



Metal 7, test4, TritonRoute-WXL



Metal 7, test4, ours

04 | Conclusion

➢ Conclusion

- An effective detailed routing algorithm with a scalable IP formulation.
- An efficient design rule checking engine which supports examining nets with multiple routes simultaneously.
- High quality candidate routes are generated with adaptive global route guides following.
- Achieves better solution quality or shorter runtime than the state-of-the-art (SOTA) results.

Thank You!

Backup

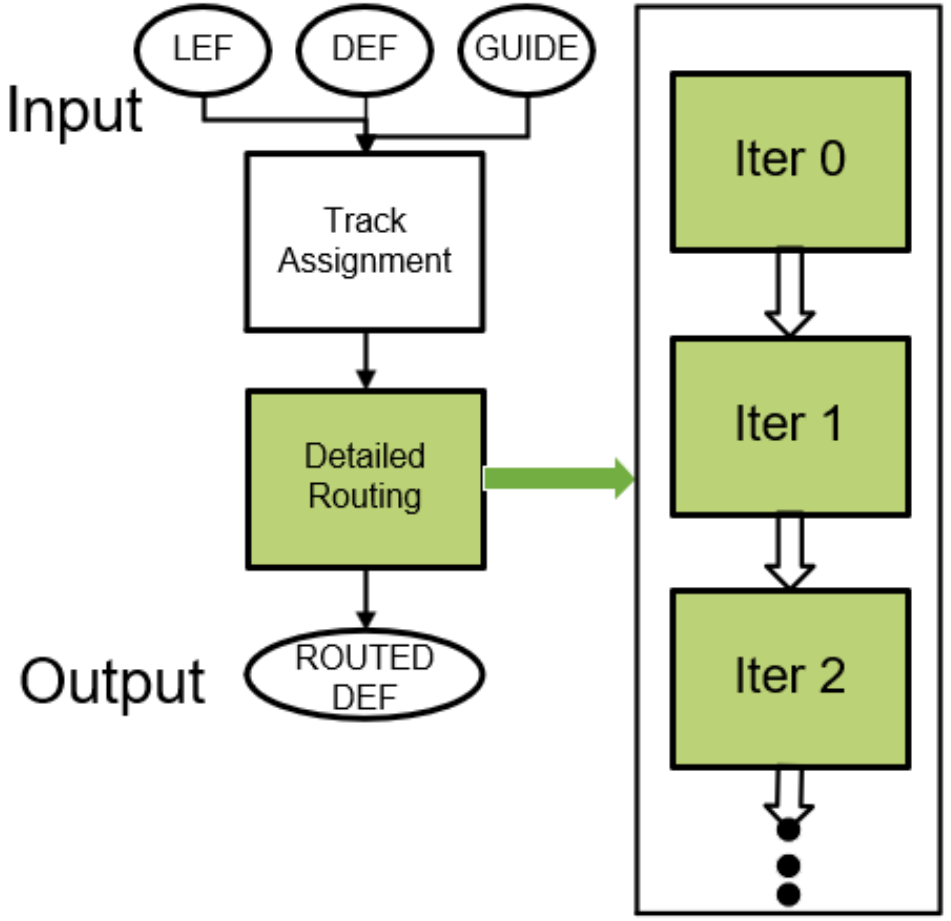
➢ Integer Programming Formulation

$$c_{it} = (wl_{it} + \alpha \cdot wl'_{it} + \beta \cdot via_{it} + \gamma \cdot drc_{it})$$

$$p_{iu,jv} = \gamma \cdot drc_{iu,jv}$$

- c_{it} is the cost of routing tree t of net n_i
- $p_{iu,jv}$ is the violations cost between routing tree u of net n_i and routing tree v of net n_j

➢ Overall Flow



- In each iteration, the entire layout is partitioned into non-overlapped rectangular regions.

➢ Our Contributions

- An effective detailed routing algorithm with a scalable IP formulation.
- An efficient design rule checking engine which supports examining nets with multiple routes simultaneously.
- High quality candidate routes are generated with adaptive global route guides following.
- Achieves better solution quality or shorter runtime than the state-of-the-art (SOTA) results.