references

1. Book Chapters and Lecture Notes
2. Papers with Code
   2.1 CRoute/RWRoute
   2.2 OpenPARF router
   2.3 VPR router
3. SAT/SMT-based
4. Parallel SSSP/Route
5. Bidirectional SSSP
6. Bidirectional PathFinder

1. Book Chapters and Lecture Notes

- M. Hutton, V. Betz, and J. Anderson, "Section 16.4.3 'Routing' in Chapter 16 'FPGA Synthesis and Physical Design'", in Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology, CRC Press 2016.
  📎Hutton, Betz, Anderson – 2016 – FPGA Synthesis and Physical Design.pdf
- Daniel Gomez–Prado, Maciej Ciesielski, "A Tutorial on FPGA Routing", in EEL4930/5934 – Reconfigurable Computing, University of Florida, Fall 2011.
  http://www.gstitt.ece.ufl.edu/courses/fall11/eel4930_5934/reading/Routing.pdf
- Philip Brisk, "FPGA Routing", in CS 223, UC Riverside,
  https://www.slideserve.com/kylene/fpga–routing

2. Papers with Code

2.1 CRoute/RWRoute

- https://github.com/UGent–HES/FPGA–CAD–Framework
- https://github.com/Xilinx/RapidWright/blob/master/src/com/xilinx/rapidwright/rwroute/PartialRouter.java

- 📎Zhou – 2022 – FPGA Placement and Routing From Academia to Industry.pdf
- 📎Zhou et al. – 2022 – RWRoute An Open-source Timing-driven Router for Commercial FPGAs.pdf
- 📎Zhou, Vercruyce, Stroobandt – 2020 – Accelerating FPGA Routing Through Algorithmic Enhancements and Connection-aware Parallelization.pdf
- 📎Vercruyce et al. – 2019 CRoute.pdf

2.2 OpenPARF router

- https://github.com/PKU-IDEA/OpenPARF/tree/master/openparf/routing
- J. Wang, J. Mai, Z. Di, and Y. Lin, "A Robust FPGA Router with Concurrent Intra-CLB Rerouting," in Proceedings of the 28th Asia and South Pacific Design Automation Conference (ASP-DAC 2023), Jan. 2023, pp. 529—534.
  - 📎Wang et al. – 2023 – A Robust FPGA Router with Concurrent Intra-CLB Rerouting.pdf
  - slides: 📎OpenPARF-router-slides.pdf

2.3 VPR router

- https://github.com/verilog-to-routing/vtr-verilog-to-routing/tree/master/vpr/src/route
- 📎Murray et al. – 2020 – VTR 8 High-performance CAD and Customizable FPGA Architecture Modelling.pdf
- 📎Murray, Zhong, Betz – 2020 – AIR A Fast but Lazy Timing-Driven FPGA Router.pdf

3. SAT/SMT-based

- https://github.com/Xilinx/RapidWright/blob/master/src/com/xilinx/rapidwright/router/SATRouter.java
- H. Fraisse and D. Gaitonde, “A SAT-based Timing Driven Place and Route Flow for Critical Soft IP,” in 2018 28th International Conference on Field Programmable Logic and Applications (FPL), Aug. 2018, pp. 8—87, doi: 10.1109/FPL.2018.00009.
- H. Fraisse, A. Joshi, D. Gaitonde, and A. Kaviani, “Boolean Satisfiability-Based Routing and Its Application to Xilinx UltraScale Clock Network,” in Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays – FPGA ’16, 2016, pp. 74—79, doi: 10.1145/2847263.2847342.

4. Parallel SSSP/Route

- Xiaojun Dong, Yan Gu, Yihan Sun, and Yunming Zhang. Efficient Stepping Algorithms and Implementations for Parallel Shortest Paths. In *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures*, pp. 184–197, 2021.
- https://github.com/ucrparlay/Parallel-SSSP

5. Bidirectional SSSP

- Vaira G, Kurasova O. "Parallel bidirectional Dijkstra's shortest path algorithm". Databases and Information Systems VI, Frontiers in Artificial Intelligence and Applications, 2011, 224: 422–435.

📎Parallel Bidirectional Dijkstra's Shortest Path Algorithm.pdf

- Tangjittaweechai L. "Parallel Shortest Path Algorithms for Graphics Processing Units." Asian Institute of Technology, 2016.

📎Parallel-Shortest-Path-Algorithms-for-Graphics-Processing-Units.pdf

6. Bidirectional PathFinder

PathFinder (simplified) Python

```python
while True: # see "while (routeIteration < config.getMaxIterations()) {...}" in routeIndirectConnections()

    for (s, t) in (s, T) in M: # see "for (Connection connection : sortedIndirectConnections) {...}" in routeIndirectConnections()
        min_heap_s.push((null, s), 0)
        min_heap_t.push((null, t), 0)
        routes = ()

        while min_heap_s.isEmpty() == False and min_heap_t.isEmpty() == False: # see "while ((rnode = queue.poll()) != null) {...}" in routeConnection(...)

            (edge_s = (prev_s, curr_s), curr_dist) = min_heap_s.pop()
            if curr_s in routes: continue
            update_present_cost(prev_s)
            routes_s.insert(edge_s)

            for next_s in fanout(curr_s): # see "for (RouteNode childRNode:rnode.getChildren()) {...}" in exploreAndExpand(...)
                if next_s in routes_s: continue
                next_dist_s = curr_dist_s + cost(curr_s) + (est(next_s, t) - est(curr_s, t))
                min_heap_s.push((curr_s, next_s), next_dist_s)

            (edge_t = (prev_t, curr_t), curr_dist_t) = min_heap_t.pop()
            if curr_t in routes_t: continue
            update_present_cost(prev_t)
            routes_t.insert(edge_t)

            for next_t in fanin(curr_t): # see "for (RouteNode childRNode:rnode.getChildren()) {...}" in exploreAndExpand(...)
                if next_t in routes_t: continue
                next_dist_t = curr_dist_t + cost(curr_t) + (est(next_t, s) - est(curr_t, s))
                min_heap_t.push((curr_t, next_t), next_dist_t)

            if |routes_s ∩ routes_t| != 0 : break # Two sets have overlap edges, means that s/t meets
                route = compare_intersect_routes(routes_s, routes_t)

    for node in route: # see updateCostFactors()
        update_history_cost(node)

    if conflicts.isEmpty(): break
```

```
    for (s, t) in (s, T) in M:
        ripup_illegal_connections()
        ripup_delay_degraded_connections()
```

Node cost: $f(n) = c_{prev} + \frac{b(n) \cdot h(n) \cdot p(n)}{share(n)} + \alpha \cdot c_{exp}$

$share(n)$: #connections that legally share the node with the connection that SSSP algorithm currently searching for

$$c_{exp} = \frac{n_{ortho} \cdot b_{ortho}}{\beta \cdot share(n)} + \frac{n_{samedir} \cdot b_{samedir}}{\beta \cdot share(n)} + b_{ipin} + \frac{b_{sink}}{\beta}$$

Maybe $\beta = 2$.