Project Title: Fast Global Routing Algorithm with Congestion Awareness

Introduction:

In the realm of VLSI physical design, global routing is a pivotal stage where approximate interconnect paths are planned before detailed routing. As chip complexity grows, routing congestion becomes a critical concern affecting timing, power, and area. This project focuses on implementing a fast global routing algorithm that is aware of congestion and adapts its routing strategy to reduce overuse of resources.

My implementation uses the A* pathfinding algorithm with a congestion-penalizing cost function, enhanced by a rip-up and reroute mechanism to iteratively resolve congested areas. I evaluated our router using grid-based benchmarks and output the routing paths, wirelength, and congestion scores.

Objective:

The primary objective of this project is to:

- Implement a global router that supports grid-based benchmarks.
- Use the A* search algorithm for efficient routing with congestion awareness.
- Apply rip-up and reroute strategies to improve congested paths.
- Evaluate the router using standard benchmarks (ISPD, ICCAD) and measure performance in terms of wirelength and congestion.

Methodology:

The routing process in this project follows a structured, iterative approach that balances pathfinding with congestion management. By combining A* search with adaptive rerouting strategies, the router efficiently connects all nets while minimizing wirelength and avoiding congested regions whenever possible. The core methodology consists of the following steps:

Step 1: Grid Initialization and Net Parsing

The input benchmark file is parsed to extract:

- Grid size, defining the routing area as a 2D grid.
- Netlist, where each net consists of a source and a sink coordinate.

An internal congestion map is initialized to track the number of times each grid cell is used during routing.

Step 2: Net Routing Using A* Search

Each net is routed using the A* pathfinding algorithm, which explores potential paths based on a cost function. The A* heuristic favors shorter paths (using Manhattan distance) but is also aware of grid congestion.

- If congestion is low, the router selects the most optimal path.
- If congestion is high near a net's path, it avoids those regions unless rerouting is explicitly triggered.

Step 3: Congestion Detection and Rerouting

After an initial routing attempt:

- The router evaluates the congestion map to identify overused grid cells.
- If any paths pass through these congested cells, they are ripped up and rerouted, taking the updated congestion into account.
- This process is repeated until all nets are routed and congestion stabilizes.

Step 4: Adaptive Relaxation

If the router can't find a valid path for a net because of heavy congestion, it temporarily loosens its restrictions and allows the net to pass through those crowded areas. This way, the algorithm makes sure that every net gets routed, even if it means some areas may still end up a bit congested.

Step 5: Routing Metrics and Output Generation

Once all nets are routed, the router computes key metrics:

- Wirelength per net and total wirelength
- Overused node count (congestion > threshold)
- Maximum and average congestion

Results:

At first, I experimented with a few test cases by setting the grid size to small and medium levels and keeping the congestion threshold below 10. This helped me observe how the router behaves under varying routing densities and how sensitive it is to congestion in tighter layouts.

For one of the initial test cases, I used a 10×10 grid with the congestion threshold set to 3. I routed three nets with varying start and end points to observe the behavior of the algorithm under moderate constraints. The results were quite promising. All nets were successfully routed with wirelengths ranging from 4 to 8 units. More importantly, the congestion remained well within limits—no nodes were overused, and the maximum congestion observed was just 2. The average congestion across the grid was only 0.17, which indicates that the paths were well distributed without causing any significant overlap. Overall, the router maintained a balanced layout, and the congestion summary confirmed that no critical paths were overly congested.

```
Routing Path for (2, 1) -> (9, 9):
(2, 1) -> (3, 2) -> (4, 3) -> (5, 4) -> (6, 5) -> (7, 6) -> (8, 7) -> (9, 8) -> (9, 9)
Total Wirelength: 8


Routing Path for (3, 2) -> (8, 7):
(3, 2) -> (4, 3) -> (5, 4) -> (6, 5) -> (7, 6) -> (8, 7)
Total Wirelength: 5


Routing Path for (5, 5) -> (9, 9):
(5, 5) -> (6, 6) -> (7, 7) -> (8, 8) -> (9, 9)
Total Wirelength: 4


[RESULT] Congestion Summary:
  - Overused Nodes: 0
  - Max Congestion: 2
  - Avg Congestion: 0.17
[STATUS] Congestion: Balanced
```

In the case of a larger grid (specifically 324×324 with 100 routing paths), I set the congestion threshold to 5. During the initial routing pass, a few nodes such as (27, 24), (12, 13), and (11, 12) were flagged as congested, and the router was unable to find valid paths for several nets that passed through them. This led to temporary warnings indicating no valid path could be found. However, once the algorithm entered the rerouting phase and allowed some flexibility, it was able to successfully find alternative paths that avoided the congested nodes. As a result, all nets were eventually routed without exceeding the congestion threshold, demonstrating the effectiveness of the adaptive rip-up and reroute strategy in resolving localized congestion.

Benchmark:

I used the ISPD 2008 global routing benchmark for evaluation. First, I converted the input files into a grid-based format compatible with my router. For large benchmarks, I set the congestion threshold to 7 to allow reasonable routing flexibility. Initially, due to the high routing density, many grid nodes became congested, resulting in several nets failing to find valid paths. However, after applying the rip-up and re-route strategy, the router was able to bypass the congested regions and successfully find alternate paths for the affected nets. The computed metrics include the wirelength of each routed path, the total wirelength for the entire benchmark, the average and maximum congestion values across the grid, and the number of overused nodes after rerouting was completed.

After evaluating several benchmarks from the ISPD 2008 global routing suite, the computed metrics are summarized in the table below.

Github Release link for benchmark Evaluation: Benchmark Evaluation

Benchmarks	**Total wirelength**	**Max Congestion**	**Avg Congestion**	**Overused nodes**
Adaptec1 3d	138305	2	0.8	0
Adaptec2 3d	253962	4	1.15	0
Newblue1 3d	133265	2	0.7	0
Newblue2 3d	195395	2	0.18	0
Bigblue1 3d	110851	2	0.33	0
Bigblue2 3d	271067	3	0.86	0
Bigblue4 3d	711873	3	0.86	0

Challenges Encountered:

One of the main challenges was achieving convergence in highly congested layouts without relying on excessive rerouting, which can significantly slow down the overall process. Balancing the congestion sensitivity in the cost function was also tricky—if the penalty was too high, the router would take unnecessarily long detours. On the other hand, if it was too low, it would allow too much overlap. Another challenge was keeping the execution time within a reasonable limit, particularly for large-scale ISPD benchmarks. As the number of nets and grid size increases, the iterative rip-up and reroute process becomes significantly more time-consuming. For example, in the case of the adaptec3 benchmark (774x779 Grid size, 1236883 routing paths), the routing process took an exceptionally long time to complete. In fact, the runtime exceeded practical limits, and I was unable to generate the final output for that benchmark. This highlights the need for further optimization, especially in handling large and densely packed designs.

Conclusion:

This project showcases a practical and effective congestion-aware global routing algorithm that uses A* search along with a rip-up and re-route strategy. The approach is both adaptive and efficient, striking a good balance between minimizing total wirelength and managing congestion across the grid. It works well for early stages of VLSI physical design, where routing decisions have a big impact on overall layout quality. The code is also modular and easy to extend, which opens the door to adding more advanced features in the future—like support for multi-pin nets, multi-layer routing, or even timing-driven optimization. Overall, this router lays a strong foundation for continued improvements and exploration in congestion-aware routing.

Reference:

Github Repository : Congestion-Aware Global Routing Algorithm

Benchmark Evaluation : Benchmark Evaluation

ISPD Benchmarks: ISPD 2008 Global Routing Benchmarks

Online Resources
Various online tutorials, blogs, and technical websites were consulted to understand A* search, global routing concepts, and congestion-aware routing strategies.

Google Colab: Google Colab

Open AI: ChatGpt