

Advanced Vlsi Project

Bitcoin

Stage 6 - Routing

שמות המגישים	אור שאול	עמרי אורן	סטאניסלאב קוגן	נועה פרקש
-----------------	----------	-----------	----------------	-----------

Workarea path for Stage6: /project/advlsi/users/orshaul/ws/bitcoin/stage6

[#P3.1_Q1]

The purpose of the routing stage in physical design is to connect the various components of a VLSI chip, such as transistors, gates, and other circuit elements, with metal wires. These connections are essential for establishing the electrical paths that enable signal transmission, power delivery, and data communication across the chip.

Here are the key objectives of the routing stage:

1. **Electrical Connectivity:** Routing ensures that all the required electrical connections between components are established according to the circuit's logical design. This includes connecting inputs and outputs of logic gates, power supply lines, and signal nets.
2. **Performance Optimization:** The routing process is crucial for optimizing the electrical performance of the chip. This involves minimizing resistance, capacitance, and inductance in the interconnections to reduce delay, power consumption, and signal integrity issues.
3. **Design Rule Compliance:** Routing must adhere to design rules set by the manufacturing process, including constraints on the width and spacing of wires, via placement, and layer usage. Compliance with these rules ensures the chip can be fabricated correctly and reliably.
4. **Area Efficiency:** Efficient routing helps minimize the area used by the interconnections, which is important for achieving a compact design. This can lead to lower costs and higher yield in manufacturing.

5. **Thermal and Power Considerations:** Routing also considers thermal management and power distribution. Proper routing of power and ground lines helps manage power distribution and thermal dissipation across the chip.
6. **Signal Integrity:** Maintaining signal integrity is crucial, especially at higher frequencies. Routing strategies aim to minimize crosstalk, electromagnetic interference, and other factors that can degrade signal quality.

Overall, the routing stage is critical for transforming a logical design into a manufacturable physical layout, ensuring that the chip functions correctly and efficiently.

[#P3.1_Q2]

The maze algorithm is a fundamental routing algorithm used in VLSI design to find paths for electrical connections between circuit components on a chip. It is especially useful for finding paths in a grid-like layout, where routing resources are discretized into a series of interconnected cells or nodes.

How the Maze Algorithm Works

The maze algorithm is essentially a pathfinding algorithm that works as follows:

1. **Grid Representation:** The chip layout is represented as a grid of cells. Each cell can either be empty (available for routing) or blocked (occupied by other routes, obstacles, or components).
2. **Initialization:**
 - The algorithm starts at a source point, which is one endpoint of the net (the connection to be routed).
 - The destination point, the other endpoint of the net, is the target to be reached.
3. **Wave Propagation (Expansion):**
 - The algorithm propagates a "wave" from the source point through the grid. This wave expands outwards, marking each visited cell with the distance (or "cost") from the source.
 - The expansion continues until the wave reaches the destination point or covers all possible paths if the destination is unreachable.

4. **Path Backtracking:**

- Once the destination is reached, the algorithm traces back from the destination to the source, following the cells with decreasing distance values. This backtracking step identifies the shortest path (or one of the shortest paths) from source to destination.

5. **Obstacle Handling:**

- The algorithm can handle obstacles by treating blocked cells as impassable. It will find an alternate path around these obstacles, if possible.

[#P3.1_Q3]

Crosstalk refers to the unwanted interference caused by electrical signals in nearby wires or components within a VLSI circuit. This phenomenon occurs due to capacitive or inductive coupling between adjacent signal lines, which can result in signal distortion, timing errors, and degraded performance of the integrated circuit.

Causes of Crosstalk

1. **Capacitive Coupling:** When two conductors are placed close to each other, a parasitic capacitance forms between them. A change in voltage in one conductor can induce a current in the neighbouring conductor, leading to crosstalk.
2. **Inductive Coupling:** As current flows through a conductor, it generates a magnetic field. If another conductor is in close proximity, this magnetic field can induce a voltage in the neighbouring conductor, contributing to crosstalk.

Effects of Crosstalk

- **Noise:** Crosstalk introduces noise into the signal, which can lead to data corruption or errors in digital circuits.
- **Delay Variations:** It can cause timing delays, impacting the speed and synchronization of the circuit.
- **Signal Integrity Issues:** Crosstalk can distort signal waveforms, affecting the accuracy of data transmission.

Minimizing Crosstalk

To minimize the effects of crosstalk, designers employ various strategies at different stages of the VLSI design process:

1. Physical Design Strategies:

- **Increased Spacing:** Increasing the distance between adjacent signal lines reduces capacitive and inductive coupling, thereby lowering crosstalk.
- **Shielding:** Sensitive or critical signal nets can be shielded by placing grounded or power traces (guard rings or shields) adjacent to them. Placing grounded guard traces or shielding lines between sensitive signal lines can reduce crosstalk by providing a path to ground for induced noise.
- **Layer Management:** Routing high-speed or sensitive signals on different layers, separated by ground or power planes, can help reduce crosstalk.

2. **Signal Integrity Techniques:**

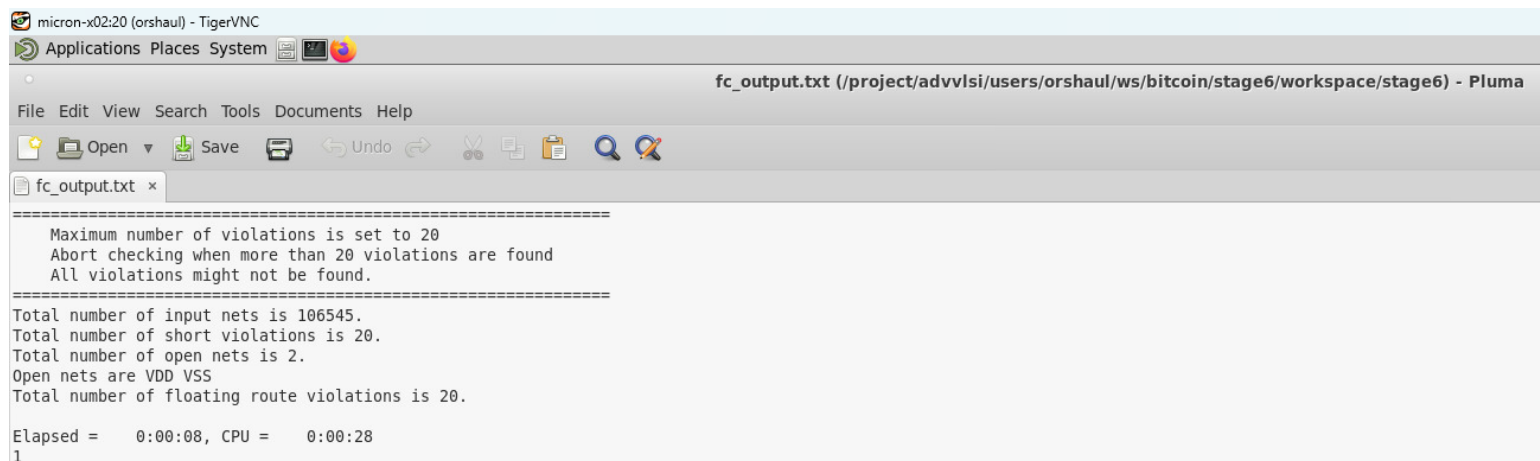
- **Signal Termination:** Properly terminating signal lines can minimize reflections and signal distortions that contribute to crosstalk.
- **Signal Integrity Analysis:** Using simulation tools to analyse and predict crosstalk effects allows designers to adjust routing and layout accordingly.

3. **Design Rule Adjustments: Adhering to Design Rules:** Following design rules specific to crosstalk minimization, such as minimum spacing requirements and maximum trace length, helps mitigate crosstalk effects.

4. **Limit Length of Parallel Nets:** Reducing the length of parallel signal lines minimizes the exposure to capacitive and inductive coupling. Shorter parallel runs reduce the area over which signals can interfere with each other, thus reducing crosstalk.

5. **Upsize Driver or Buffer:** Increasing the size of the drivers or buffers connected to a signal line can help combat the effects of crosstalk. A stronger driver or buffer can drive the signal with more power, making it more resistant to interference. However, this approach must be used carefully, as it can also increase power consumption and signal switching noise.

[#P3.2_Q1]



```
micron-x02:20 (orshaul) - TigerVNC
Applications Places System
fc_output.txt (/project/advvisi/users/orshaul/ws/bitcoin/stage6/workspace/stage6) - Pluma
File Edit View Search Tools Documents Help
Open Save Undo
fc_output.txt x
=====
Maximum number of violations is set to 20
Abort checking when more than 20 violations are found
All violations might not be found.
=====
Total number of input nets is 106545.
Total number of short violations is 20.
Total number of open nets is 2.
Open nets are VDD VSS
Total number of floating route violations is 20.
Elapsed = 0:00:08, CPU = 0:00:28
1
```

20 Short were found in our design

2 Opens were found in our design

[#P3.2_Q2]

Recommendations for Improving the Current Implementation

To address the issues of shorts and opens and to enhance the overall design, consider the following recommendations:

1. Design Rule Verification:

- **Strict Adherence to Design Rules:** Ensure that the layout follows all design rules provided by the foundry or specified in the design guidelines. This includes proper spacing, minimum feature sizes, and layer usage.

- **Automated DRC:** Regularly run automated DRC checks throughout the design process to catch and correct shorts and opens early. This can prevent issues from propagating into later stages of the design.

2. Routing Optimization:

- **Improve Routing Algorithms:** Use routing tools that optimize for minimal crossovers and avoid narrow spaces where shorts might occur. Consider using more advanced routing options that can automatically adjust to prevent shorts.
- **Manual Inspection:** In critical areas, manually inspect and adjust the routing to ensure that no unintentional connections are made and that all intended connections are properly established.

3. Layer Management:

- **Layer Allocation:** Use appropriate metal layers for different types of signals, separating high-speed or critical signals from others to minimize interference. This can also help reduce the chances of shorts by keeping sensitive lines away from noisy ones.
- **Layer Verification:** Verify that the correct layers are used according to the design specifications and that there are no layer mismatches or omissions that could lead to opens or shorts.

4. Connectivity Verification:

- **Netlist Comparison:** Perform a netlist comparison between the schematic and the layout to ensure all connections are correctly implemented. This process helps to catch missing connections (opens) or unintended ones (shorts).

5. Iterative Testing and Refinement:

- **Incremental Design Testing:** Test the design incrementally, checking each part of the layout before moving on to the next. This approach helps isolate issues and correct them before they become widespread.
- **Feedback Loop:** Create a feedback loop where issues found in DRC and other checks are communicated back to the design team for prompt resolution.