Chapter 5: Physical Design and Optimization

Algorithms for Floorplanning, Placement, and Routing

5.1 **Introduction to Physical Design in VLSI**

Physical design is the final step in the VLSI design flow, where logical representations of the circuit are transformed into a physical layout that can be fabricated. This stage involves critical tasks such as floorplanning, placement, routing, and optimization to meet performance, area, and power constraints. The goal of physical design is to efficiently use available silicon real estate while ensuring that the design meets the required functional specifications, performance metrics, and manufacturability requirements.

This chapter provides an in-depth study of the key algorithms used in the physical design process, focusing on floorplanning, placement, and routing, which are the three main stages of physical design optimization.

5.2 **Floorplanning in VLSI Design**

Floorplanning is the process of determining the relative positions of the various blocks or modules of a chip to optimize area usage and minimize interconnect delays. The primary goal of floorplanning is to create an initial layout that satisfies the design's area constraints and optimizes the placement of blocks to minimize wirelength and power consumption.

Key objectives in floorplanning include:

- **Block Placement**: Determining the relative positions of functional blocks to minimize area and wirelength.
- **Wirelength Minimization**: Minimizing the total length of the interconnects between blocks.
- **Timing Optimization**: Ensuring that critical paths are not too long to avoid timing issues.

Floorplanning algorithms can be broadly classified into two categories:

- **Analytical Algorithms**: These algorithms use mathematical models to optimize the floorplan. They are often based on concepts from operations research and use techniques like simulated annealing or force-directed methods.
- **Partitioning Algorithms**: These algorithms divide the chip into smaller partitions and then place these partitions in the most efficient manner. One of the most well-known partitioning algorithms is **Kernighan-Lin**.

5.2.1 **Floorplanning Algorithms**

- **Simulated Annealing**: This is a heuristic optimization method inspired by the annealing process in metallurgy. It gradually adjusts the positions of blocks on the chip to minimize a cost function that considers wirelength and area. By iterating through possible solutions and accepting worse solutions with a decreasing probability, simulated annealing can escape local minima to find near-optimal solutions.

- **Force-Directed Algorithms**: These algorithms simulate the forces between blocks (attractive and repulsive forces) to determine their optimal positions. They iteratively adjust the blocks' positions by considering both the area and wirelength minimization.

- **Kernighan-Lin Algorithm**: This algorithm is used for partitioning the floorplan. It works by iteratively swapping blocks between partitions to minimize the cut size (i.e., the number of interconnections between blocks in different partitions).

5.3 **Placement in VLSI Design**

Placement is the process of assigning the positions of standard cells or blocks on the chip after floorplanning. The primary objective is to minimize the total wirelength and meet timing constraints, while also ensuring the design fits within the chip's available area.

Placement algorithms aim to optimize:

- **Wirelength Minimization**: By placing related cells closer together, the total wirelength can be reduced.

- **Timing Optimization**: Critical paths must be considered during placement to avoid timing violations.

- **Density Control**: Ensuring that the placement doesn’t create areas of high congestion, which could lead to manufacturing issues.

Placement algorithms can be divided into two categories:

- **Global Placement**: Involves the initial placement of cells to roughly distribute them across the chip to minimize wirelength.

- **Detailed Placement**: Fine-tunes the cell placement by considering specific manufacturing constraints like cell overlap and routing congestion.

5.3.1 **Placement Algorithms**

- **Simulated Annealing (SA)**: Similar to its use in floorplanning, simulated annealing is also widely used in placement algorithms. It starts with a random placement and iteratively improves the placement by minimizing a cost function (e.g., wirelength, timing, or congestion). The temperature parameter in simulated annealing allows the algorithm to escape local minima, leading to better solutions.

- **Greedy Algorithms**: Greedy placement algorithms iteratively place cells by selecting the best position for each cell based on a given cost metric, such as the shortest wirelength to other cells. Although faster than simulated annealing, greedy algorithms are often not as effective at finding the global optimum.

- **Quadratic Programming**: This method is used to solve placement problems in which the objective is to minimize a quadratic function (e.g., wirelength) subject to certain constraints (e.g., timing). It's an optimization-based approach that provides high-quality solutions.

- **Partitioning-Based Placement**: These algorithms divide the layout into manageable partitions and perform placement within each partition. The partitions are then refined iteratively, and this approach helps to reduce congestion and wirelength.

5.4 **Routing in VLSI Design**

Routing is the process of connecting the cells or blocks placed in the previous steps with metal layers to form the complete circuit. The goal is to ensure that all connections are made while minimizing the total routing length, reducing power consumption, and avoiding timing violations.

Routing involves:

- **Global Routing**: Identifying the best general routes for each signal without considering the exact details of the layout.

- **Detailed Routing**: Determining the exact path for each wire, ensuring there is no overlap or interference between wires.

5.4.1 **Routing Algorithms**

- **Maze Routing**: This is a classical routing algorithm used for finding the shortest path between two points while avoiding obstacles. It works by exploring all possible paths from the source to the destination in a grid-based layout, using a breadth-first search algorithm to find the optimal route.

- **Lee's Algorithm**: This is a variation of maze routing, which uses a wave propagation technique to explore the grid and find the shortest path. It is widely used in early routing

steps and is efficient for small designs.

- *A Algorithm**: The A* algorithm is a well-known pathfinding algorithm that finds the shortest path by evaluating both the cost to reach the current point and the estimated cost to reach the destination. It's used for both global and detailed routing in VLSI designs, balancing between performance and computational efficiency.

- **Global Routing with Steiner Trees**: In this method, global routing aims to minimize the total wirelength by using Steiner trees, which are more efficient than using simple shortest-path algorithms like maze routing. Steiner trees involve adding extra "helper" points (called Steiner points) to improve routing efficiency.

5.5 **Optimization Techniques in Routing**

Optimizing routing is essential to achieve compact, efficient, and manufacturable designs. Routing optimization techniques aim to minimize wirelength, reduce signal delay, and avoid congestion.

- **Congestion Avoidance**: Algorithms focus on ensuring that there is no congestion in the routing layers. Routing congestion can lead to delays, increased power consumption, and manufacturing defects.

- **Layer Assignment**: In multi-layer designs, layer assignment algorithms determine which metal layer to use for each wire, aiming to minimize wire crossings and congestion while meeting design rules.

- **Timing-Driven Routing**: In timing-driven routing, the algorithm ensures that critical signals are routed with lower delays. This involves prioritizing certain paths that are critical to the circuit's performance.

5.6 **Conclusion**

In this chapter, we explored the key algorithms used in physical design, focusing on floorplanning, placement, and routing. These optimization techniques are essential for transforming logical designs into manufacturable, efficient, and high-performance physical layouts. As VLSI designs become more complex, advanced algorithms and optimization methods are required to meet the stringent performance, area, and power requirements of modern integrated circuits. In the following chapters, we will discuss these techniques in more detail and examine how industry-standard tools apply them in real-world designs.