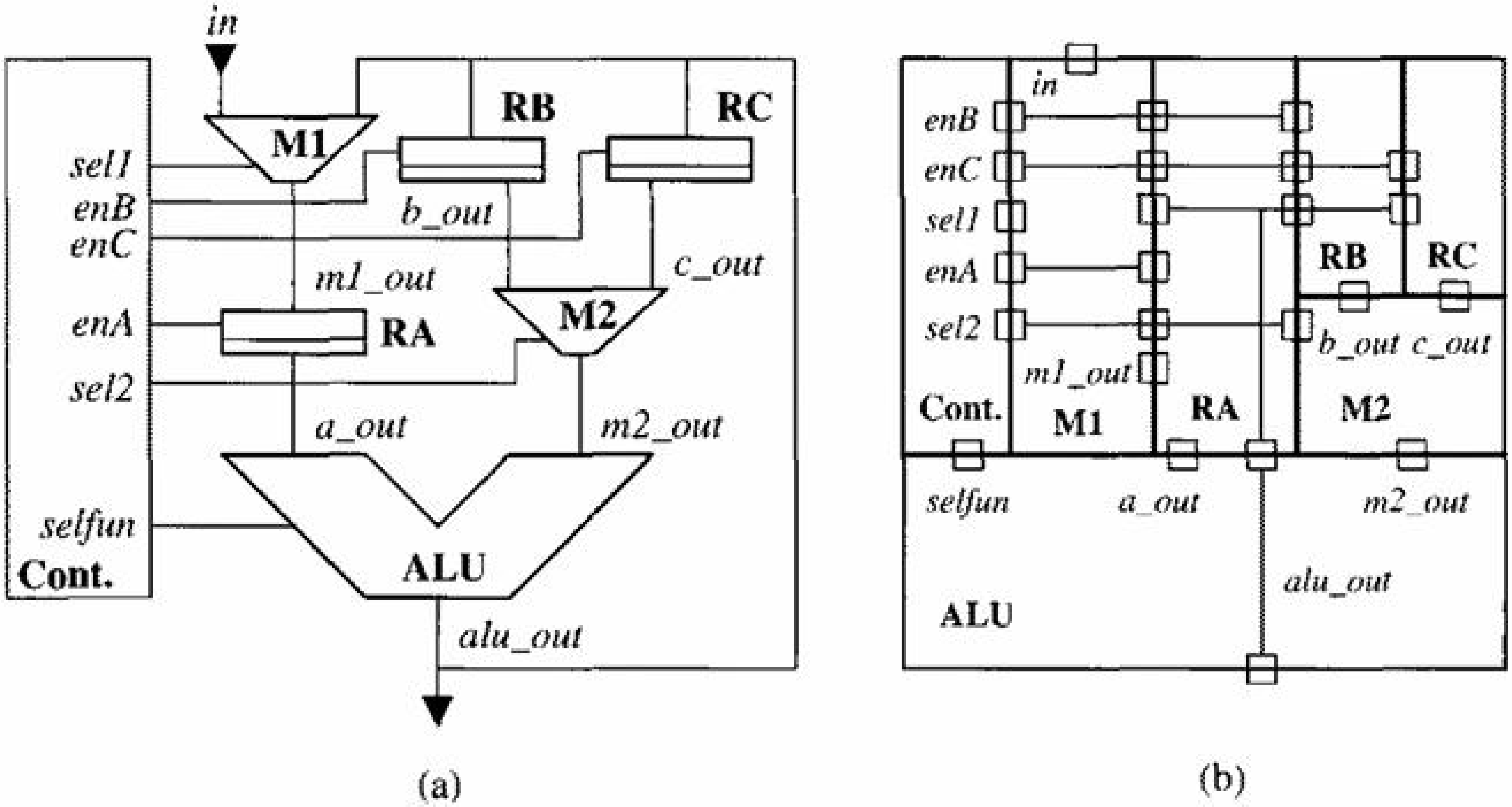
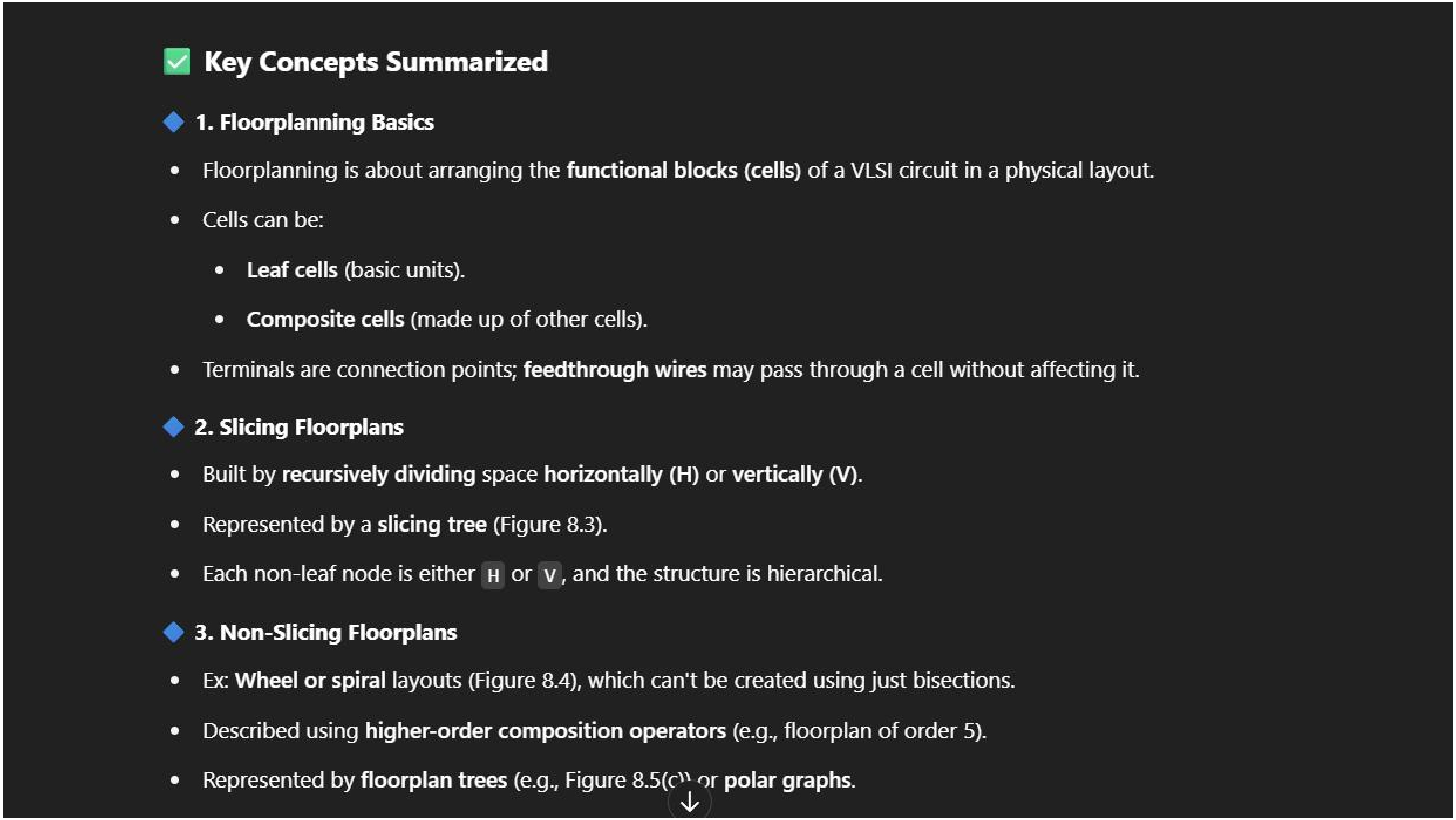


Figure 8.2 A structural description of some circuit (a) and a possible floorplan (b).



✅ Key Concepts Summarized

🔷 1. Floorplanning Basics

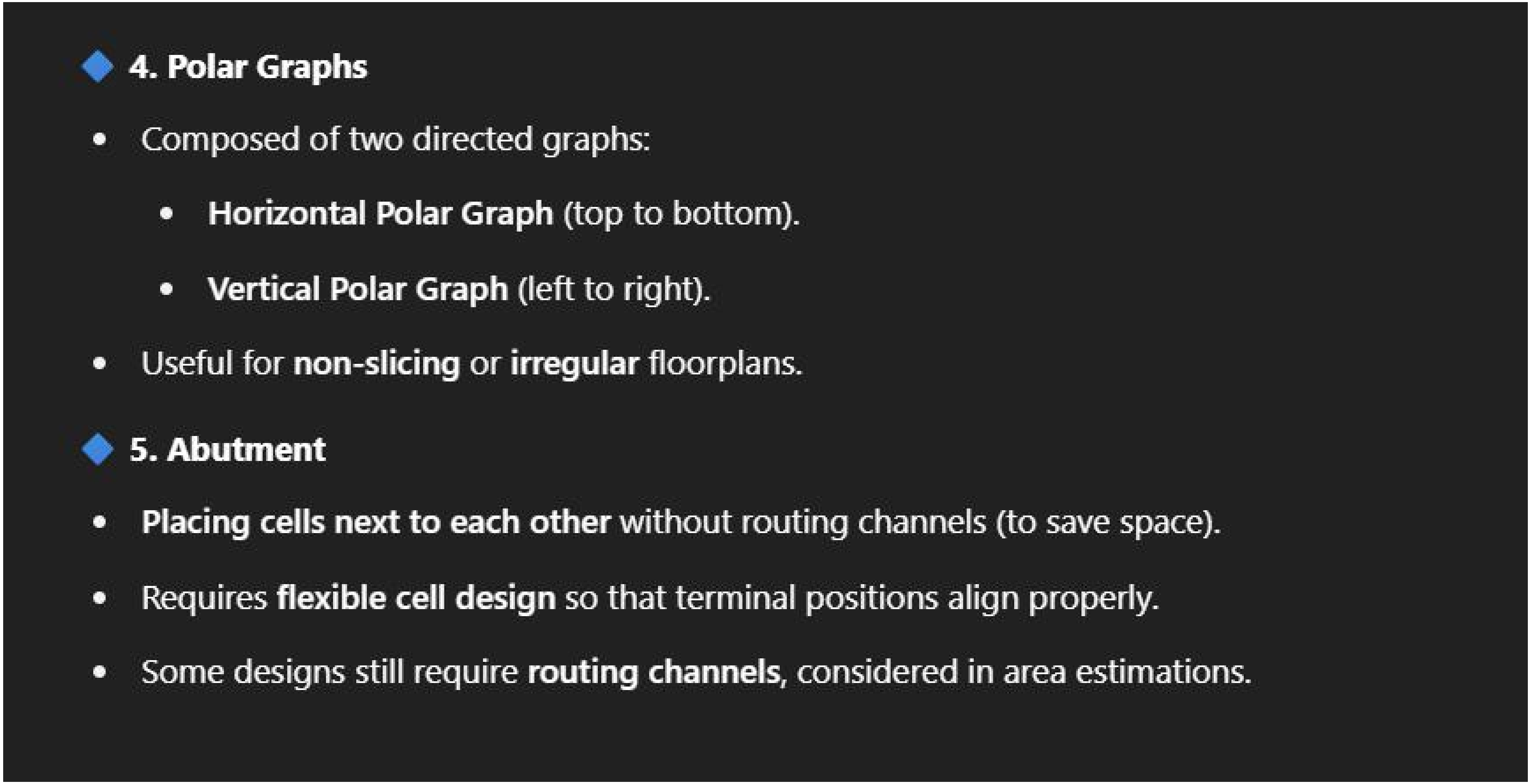
- Floorplanning is about arranging the **functional blocks (cells)** of a VLSI circuit in a physical layout.
- Cells can be:
  - **Leaf cells** (basic units).
  - **Composite cells** (made up of other cells).
- Terminals are connection points; **feedthrough wires** may pass through a cell without affecting it.

🔷 2. Slicing Floorplans

- Built by **recursively dividing** space **horizontally (H)** or **vertically (V)**.
- Represented by a **slicing tree** (Figure 8.3).
- Each non-leaf node is either `H` or `V`, and the structure is hierarchical.

🔷 3. Non-Slicing Floorplans

- Ex: **Wheel or spiral** layouts (Figure 8.4), which can't be created using just bisections.
- Described using **higher-order composition operators** (e.g., floorplan of order 5).
- Represented by **floorplan trees** (e.g., Figure 8.5(c)) or **polar graphs**.

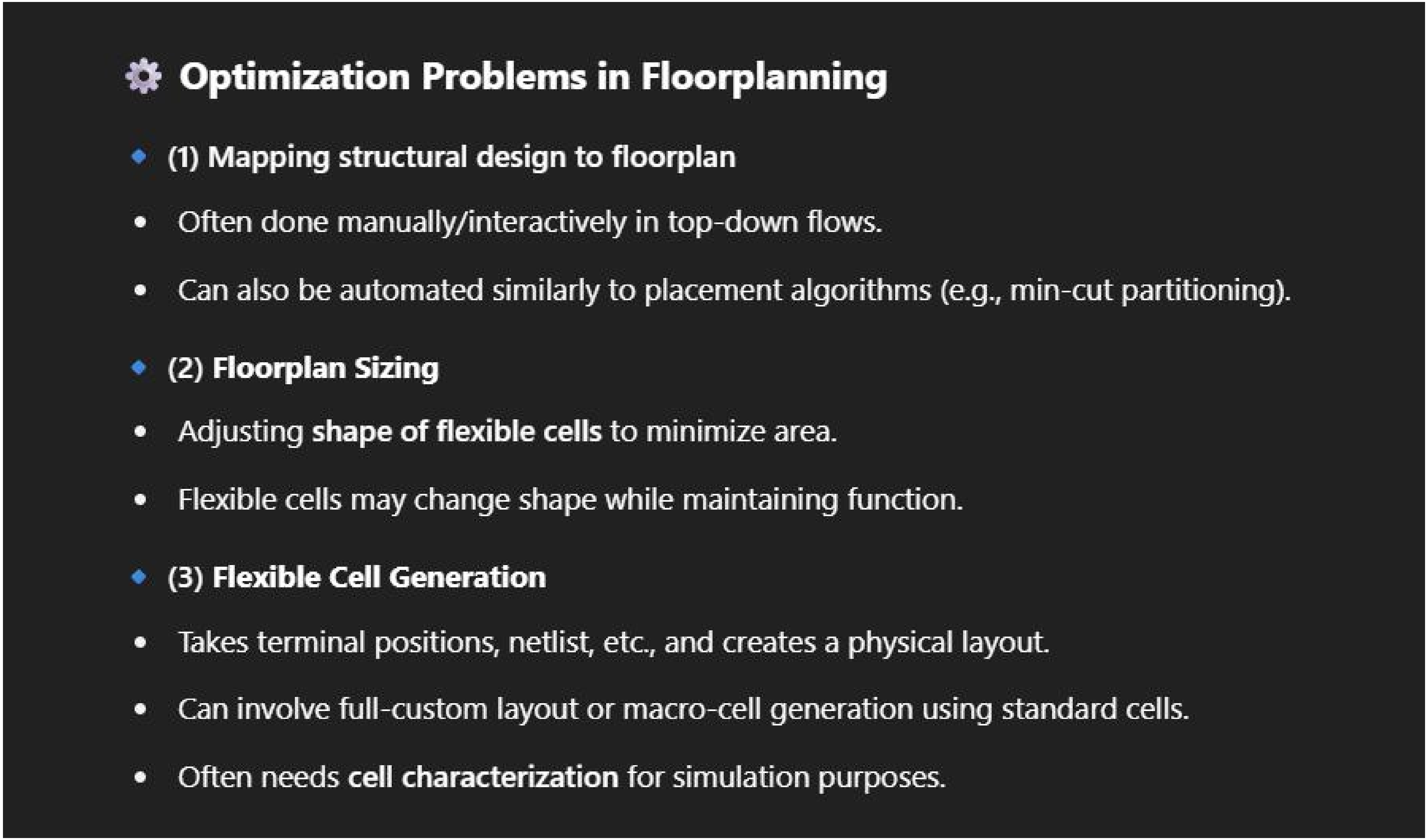


🔷 4. Polar Graphs

- Composed of two directed graphs:
  - **Horizontal Polar Graph** (top to bottom).
  - **Vertical Polar Graph** (left to right).
- Useful for **non-slicing** or **irregular** floorplans.

🔷 5. Abutment

- **Placing cells next to each other** without routing channels (to save space).
- Requires **flexible cell design** so that terminal positions align properly.
- Some designs still require **routing channels**, considered in area estimations.



⚙ Optimization Problems in Floorplanning

◆ **(1) Mapping structural design to floorplan**

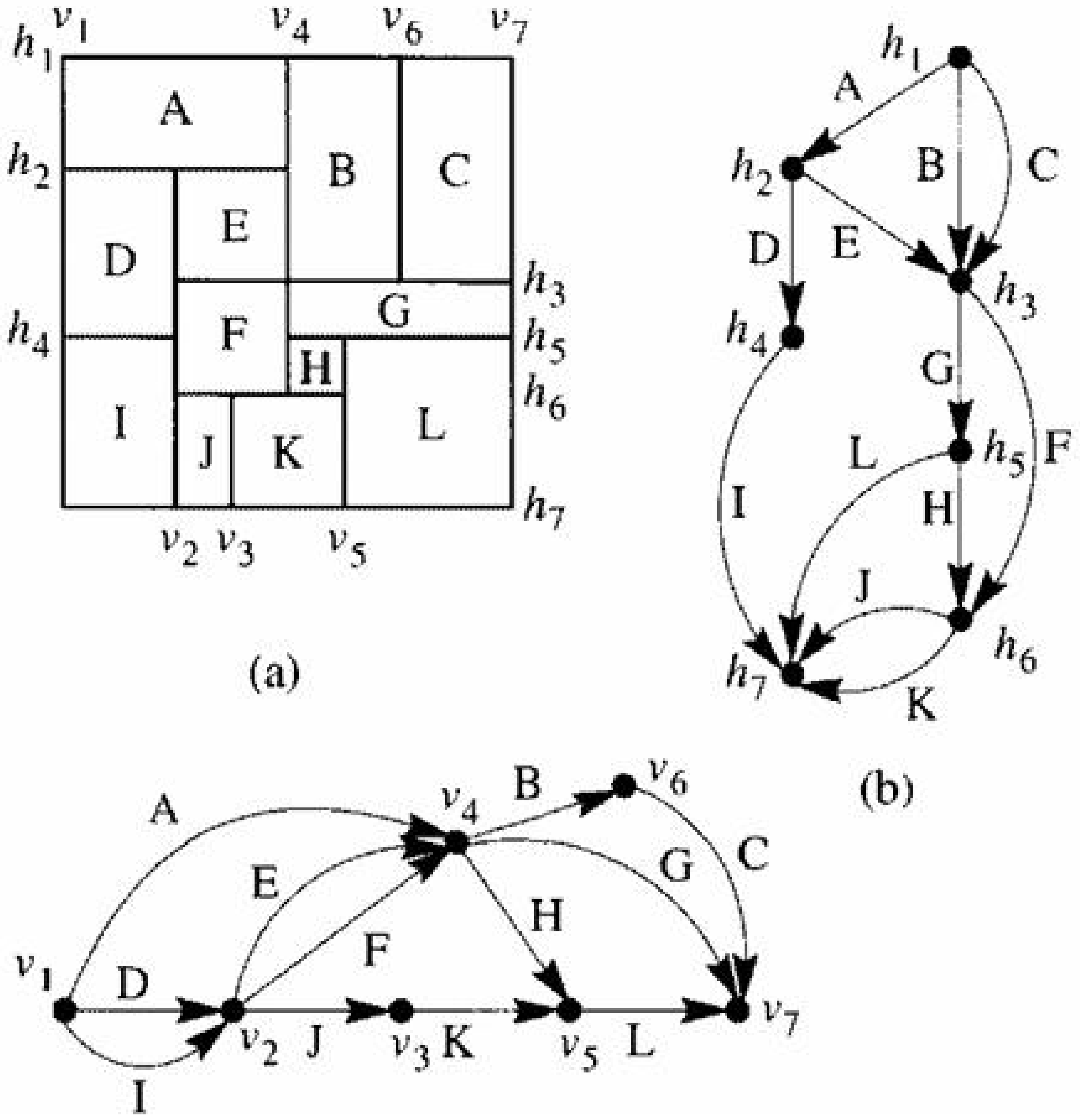
- Often done manually/interactively in top-down flows.
- Can also be automated similarly to placement algorithms (e.g., min-cut partitioning).

◆ (2) **Floorplan Sizing**

- Adjusting **shape of flexible cells** to minimize area.
- Flexible cells may change shape while maintaining function.

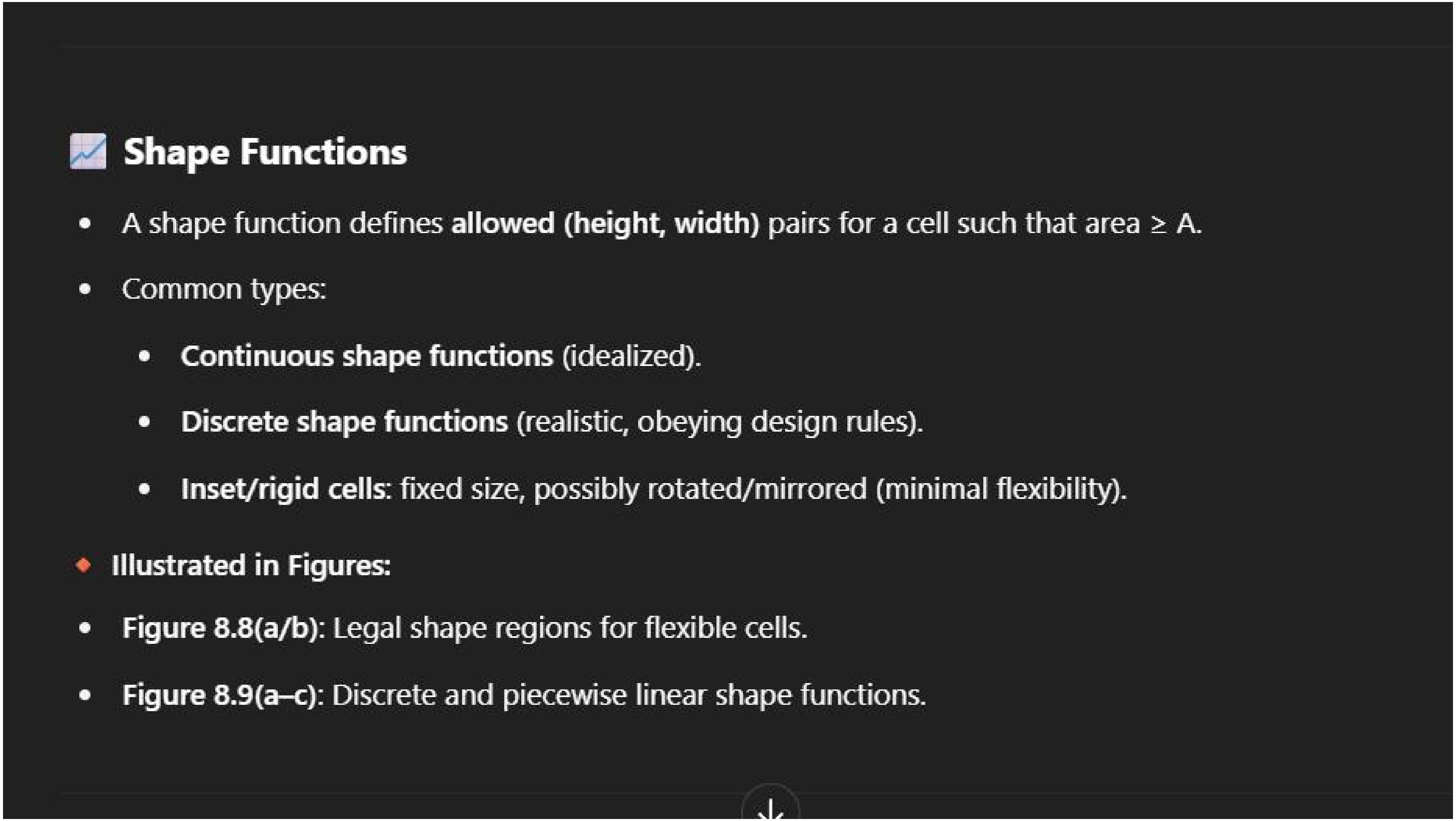
◆ (3) **Flexible Cell Generation**

- Takes terminal positions, netlist, etc., and creates a physical layout.
- Can involve full-custom layout or macro-cell generation using standard cells.
- Often needs **cell characterization** for simulation purposes.



(a)

(b)



📈 Shape Functions

- A shape function defines **allowed (height, width)** pairs for a cell such that area ≥ A.
- Common types:
  - **Continuous shape functions** (idealized).
  - **Discrete shape functions** (realistic, obeying design rules).
  - **Inset/rigid cells**: fixed size, possibly rotated/mirrored (minimal flexibility).

🔸 **Illustrated in Figures:**

- **Figure 8.8(a/b)**: Legal shape regions for flexible cells.
- **Figure 8.9(a–c)**: Discrete and piecewise linear shape functions.

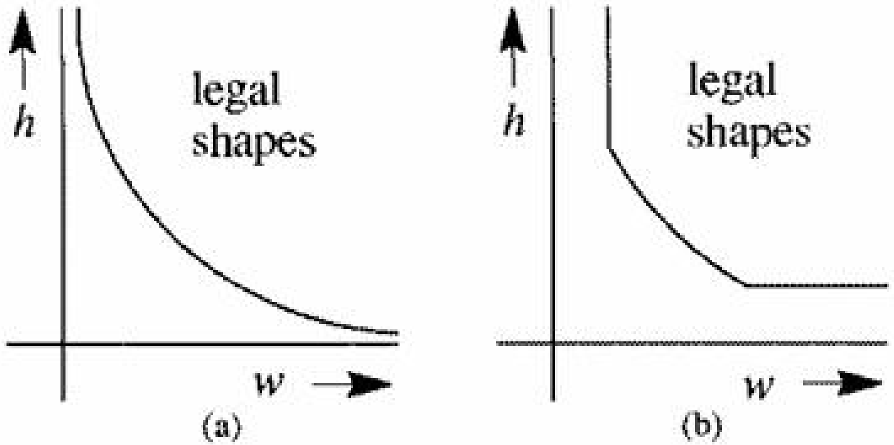


Figure 8.8 Shape functions for a cell without (a) and with (b) minimal width/height restrictions.

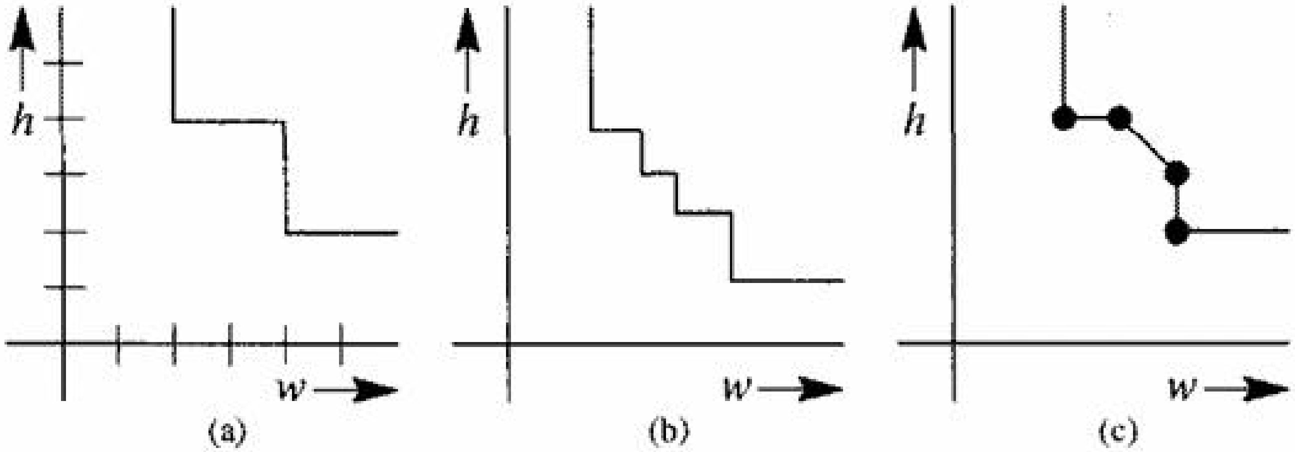
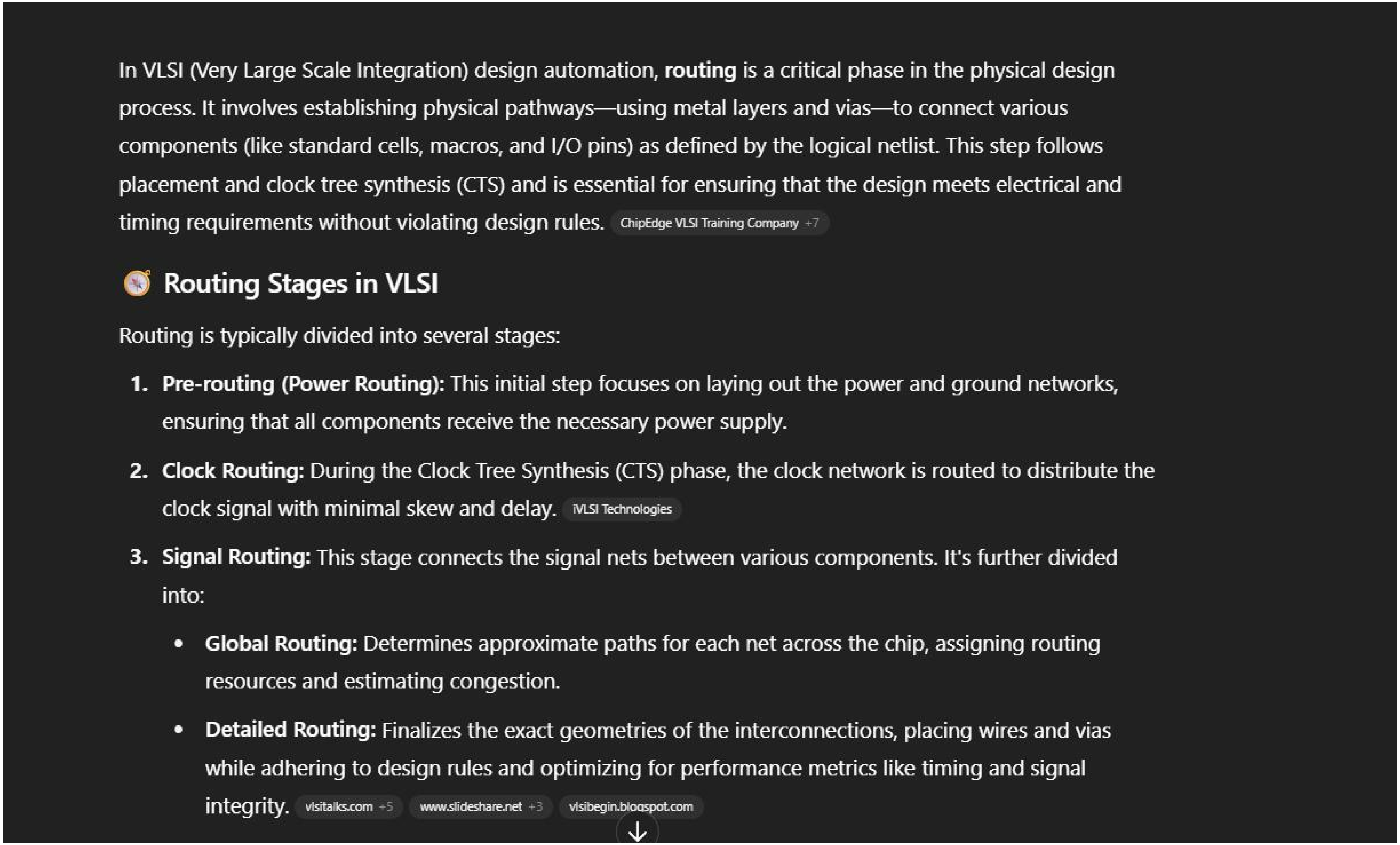


Figure 8.9 The shape function of an inset cell (a); the shape function of a cell with a discrete set of (h, w) pairs (b); a piecewise linear shape function (c).

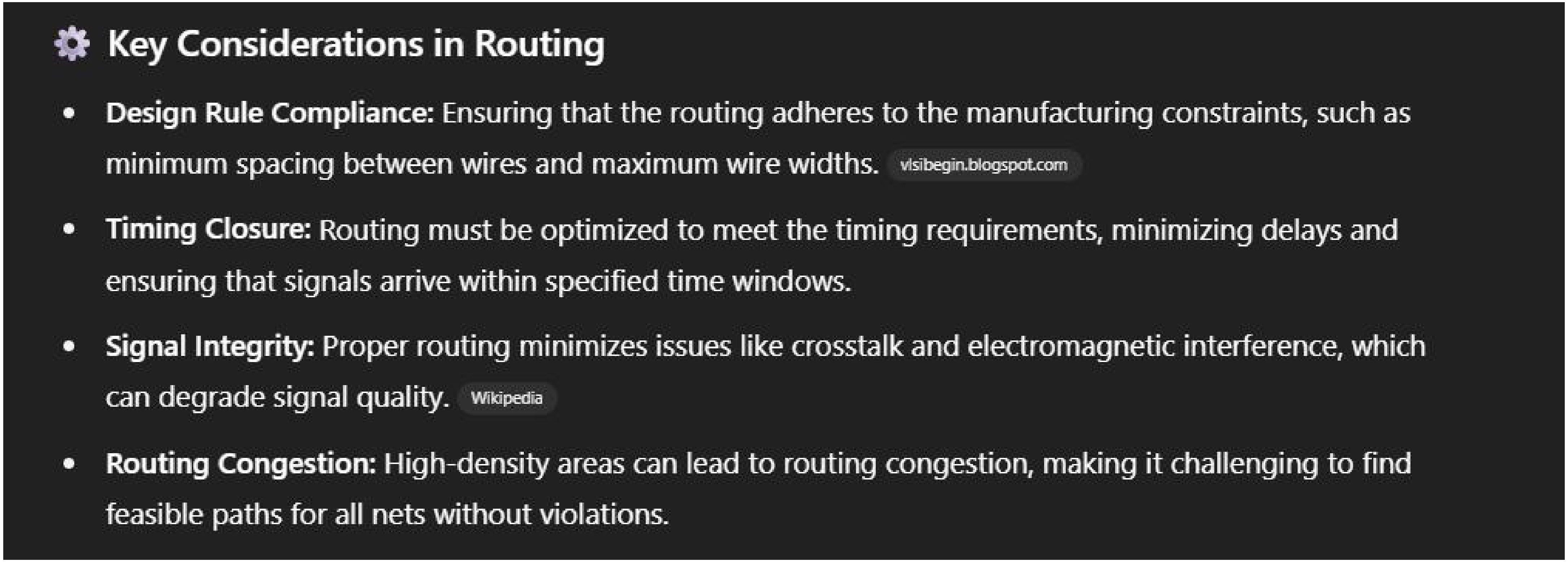


In VLSI (Very Large Scale Integration) design automation, **routing** is a critical phase in the physical design process. It involves establishing physical pathways—using metal layers and vias—to connect various components (like standard cells, macros, and I/O pins) as defined by the logical netlist. This step follows placement and clock tree synthesis (CTS) and is essential for ensuring that the design meets electrical and timing requirements without violating design rules. ChipEdge VLSI Training Company +7

🧭 Routing Stages in VLSI

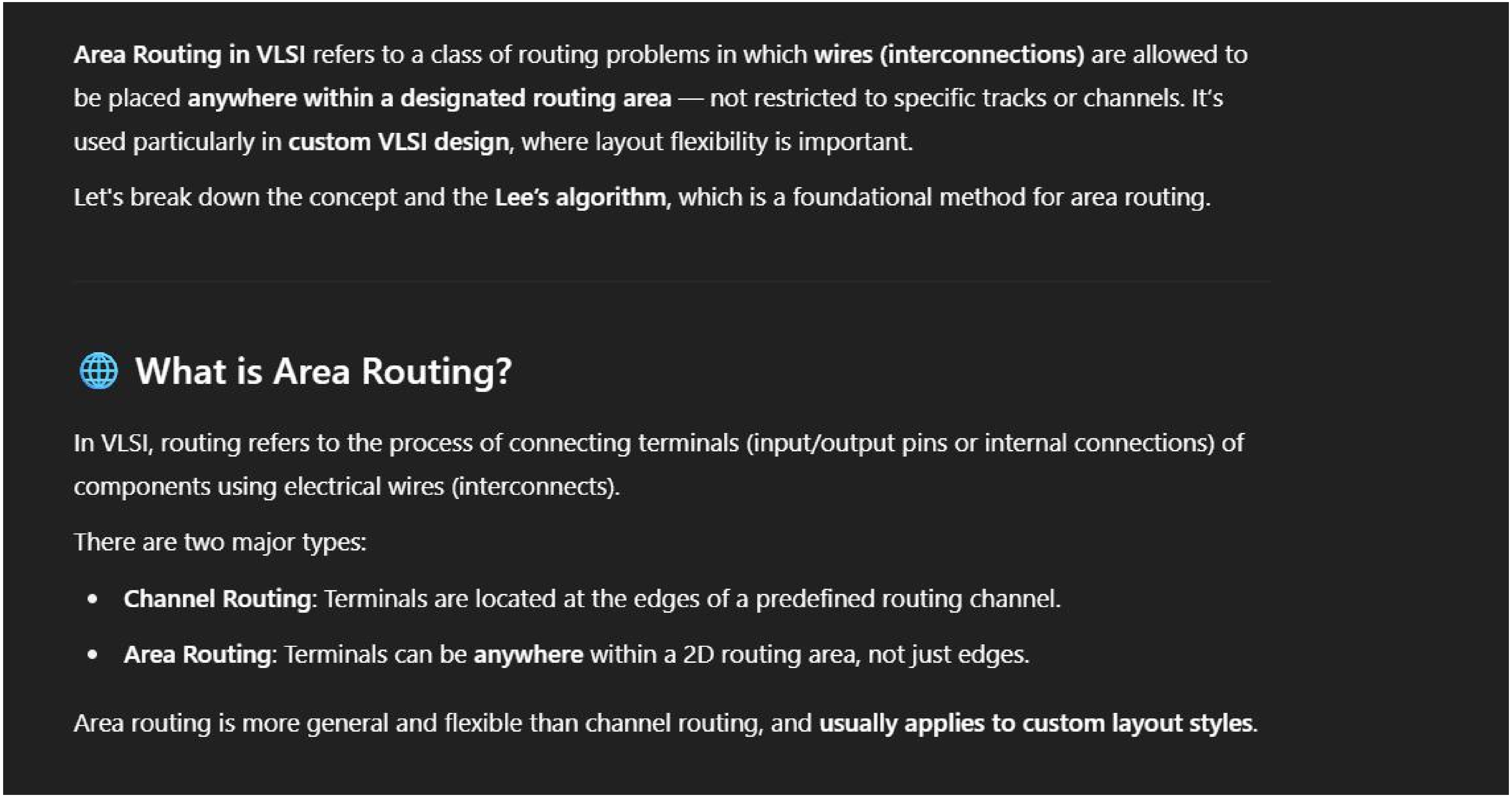
Routing is typically divided into several stages:

1. **Pre-routing (Power Routing):** This initial step focuses on laying out the power and ground networks, ensuring that all components receive the necessary power supply.
2. **Clock Routing:** During the Clock Tree Synthesis (CTS) phase, the clock network is routed to distribute the clock signal with minimal skew and delay. iVLSI Technologies
3. **Signal Routing:** This stage connects the signal nets between various components. It's further divided into:
   - **Global Routing:** Determines approximate paths for each net across the chip, assigning routing resources and estimating congestion.
   - **Detailed Routing:** Finalizes the exact geometries of the interconnections, placing wires and vias while adhering to design rules and optimizing for performance metrics like timing and signal integrity. vlsitalks.com +5 www.slideshare.net +3 vlsibegin.blogspot.com



⚙️ Key Considerations in Routing

- **Design Rule Compliance:** Ensuring that the routing adheres to the manufacturing constraints, such as minimum spacing between wires and maximum wire widths. vlsibegin.blogspot.com
- **Timing Closure:** Routing must be optimized to meet the timing requirements, minimizing delays and ensuring that signals arrive within specified time windows.
- **Signal Integrity:** Proper routing minimizes issues like crosstalk and electromagnetic interference, which can degrade signal quality. Wikipedia
- **Routing Congestion:** High-density areas can lead to routing congestion, making it challenging to find feasible paths for all nets without violations.



Area Routing in VLSI refers to a class of routing problems in which **wires (interconnections)** are allowed to be placed **anywhere within a designated routing area** — not restricted to specific tracks or channels. It's used particularly in **custom VLSI design**, where layout flexibility is important.

Let's break down the concept and the **Lee's algorithm**, which is a foundational method for area routing.

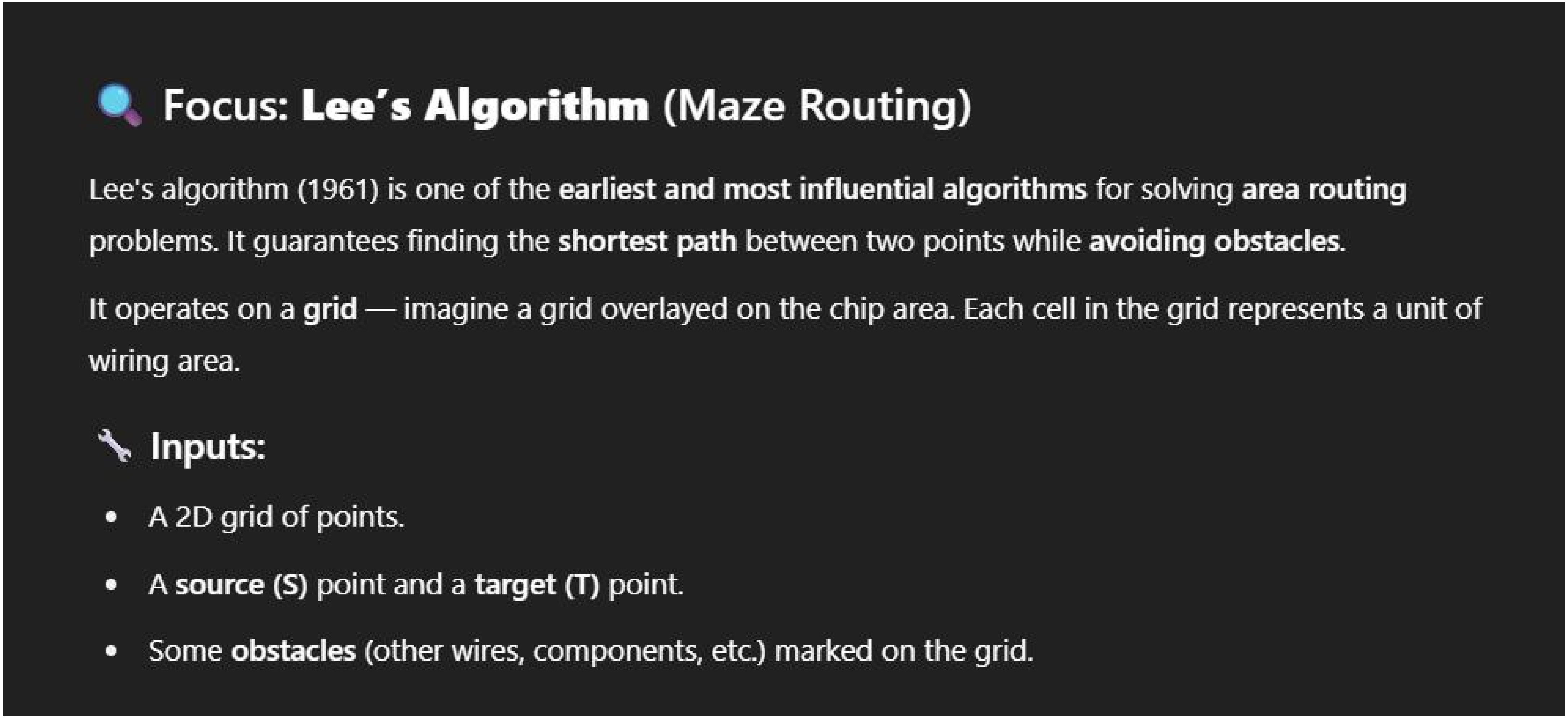
🌐 What is Area Routing?

In VLSI, routing refers to the process of connecting terminals (input/output pins or internal connections) of components using electrical wires (interconnects).

There are two major types:

- **Channel Routing**: Terminals are located at the edges of a predefined routing channel.
- **Area Routing**: Terminals can be **anywhere** within a 2D routing area, not just edges.

Area routing is more general and flexible than channel routing, and **usually applies to custom layout styles**.



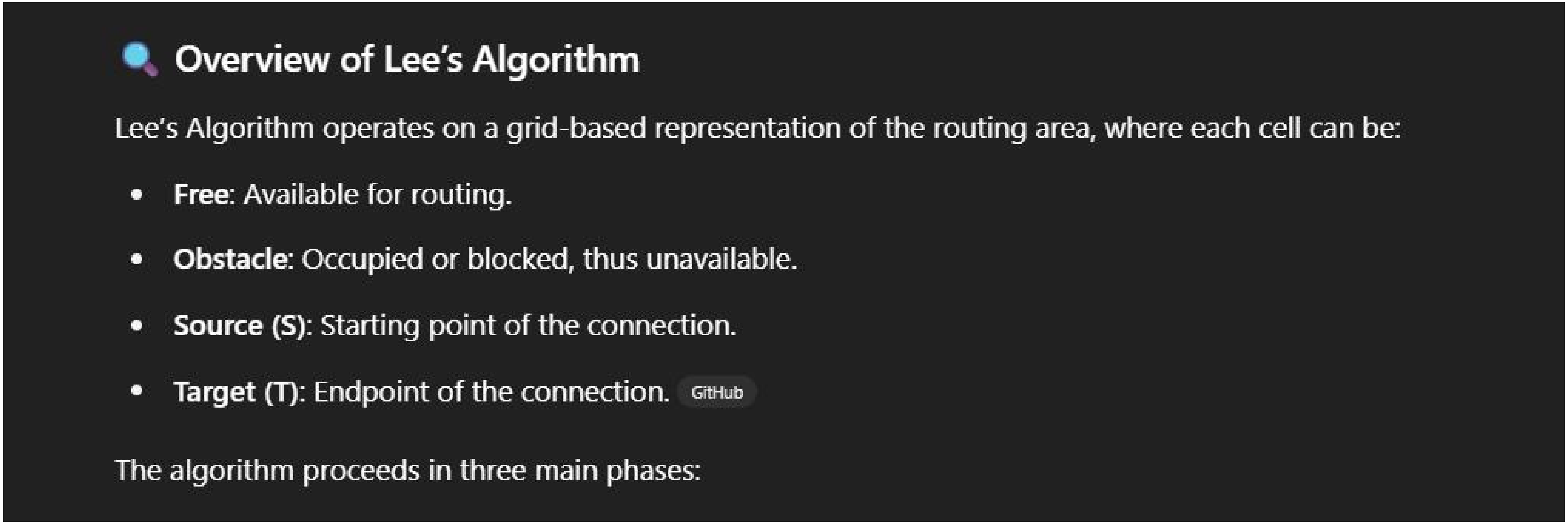
🔍 Focus: **Lee's Algorithm** (Maze Routing)

Lee's algorithm (1961) is one of the **earliest and most influential algorithms** for solving **area routing** problems. It guarantees finding the **shortest path** between two points while **avoiding obstacles**.

It operates on a **grid** — imagine a grid overlayed on the chip area. Each cell in the grid represents a unit of wiring area.

🔧 Inputs:

- A 2D grid of points.
- A **source (S)** point and a **target (T)** point.
- Some **obstacles** (other wires, components, etc.) marked on the grid.

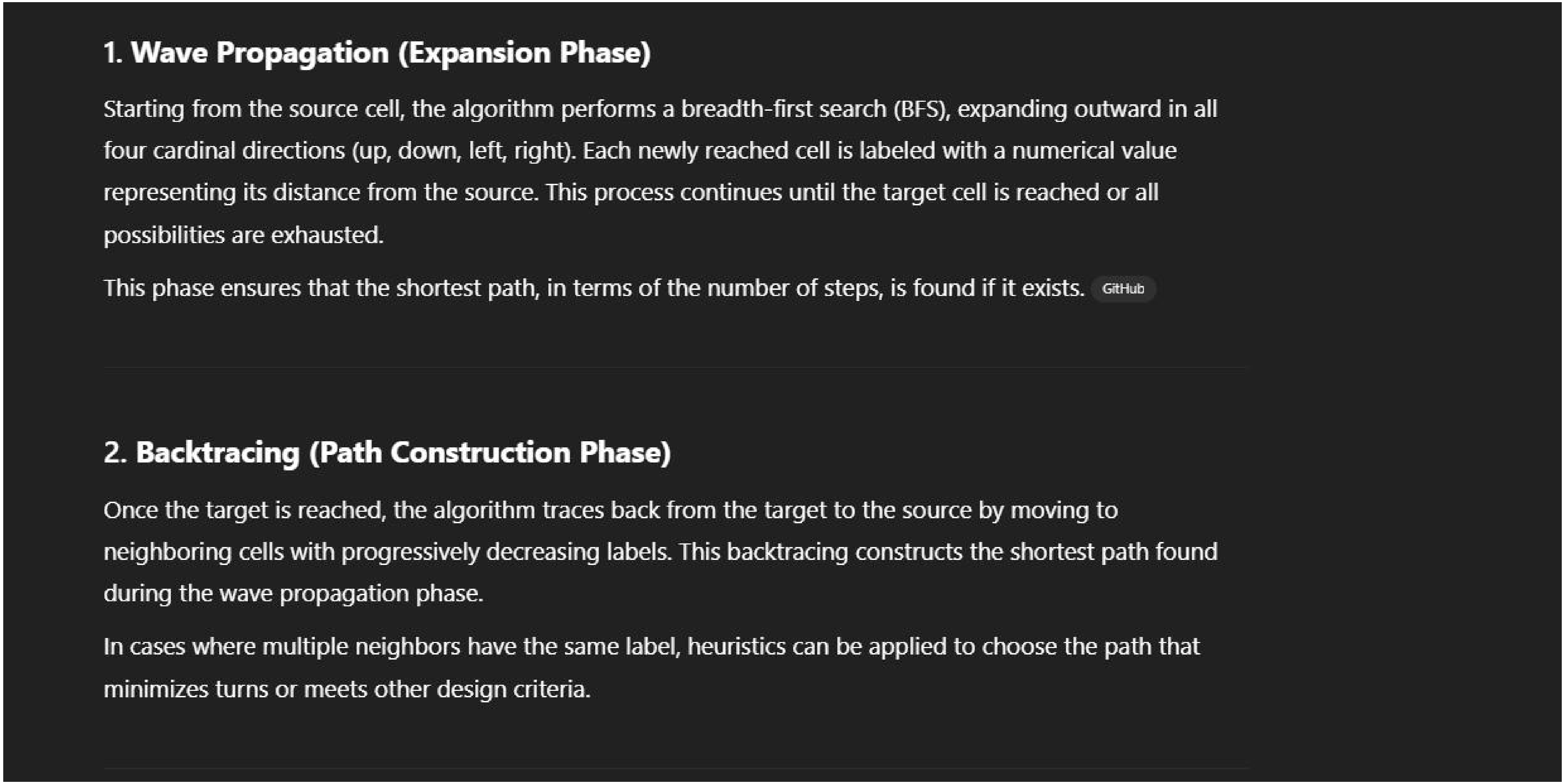


🔍 Overview of Lee’s Algorithm

Lee’s Algorithm operates on a grid-based representation of the routing area, where each cell can be:

- **Free**: Available for routing.
- **Obstacle:** Occupied or blocked, thus unavailable.
- **Source (S):** Starting point of the connection.
- **Target (T):** Endpoint of the connection. GitHub

The algorithm proceeds in three main phases:



1. Wave Propagation (Expansion Phase)

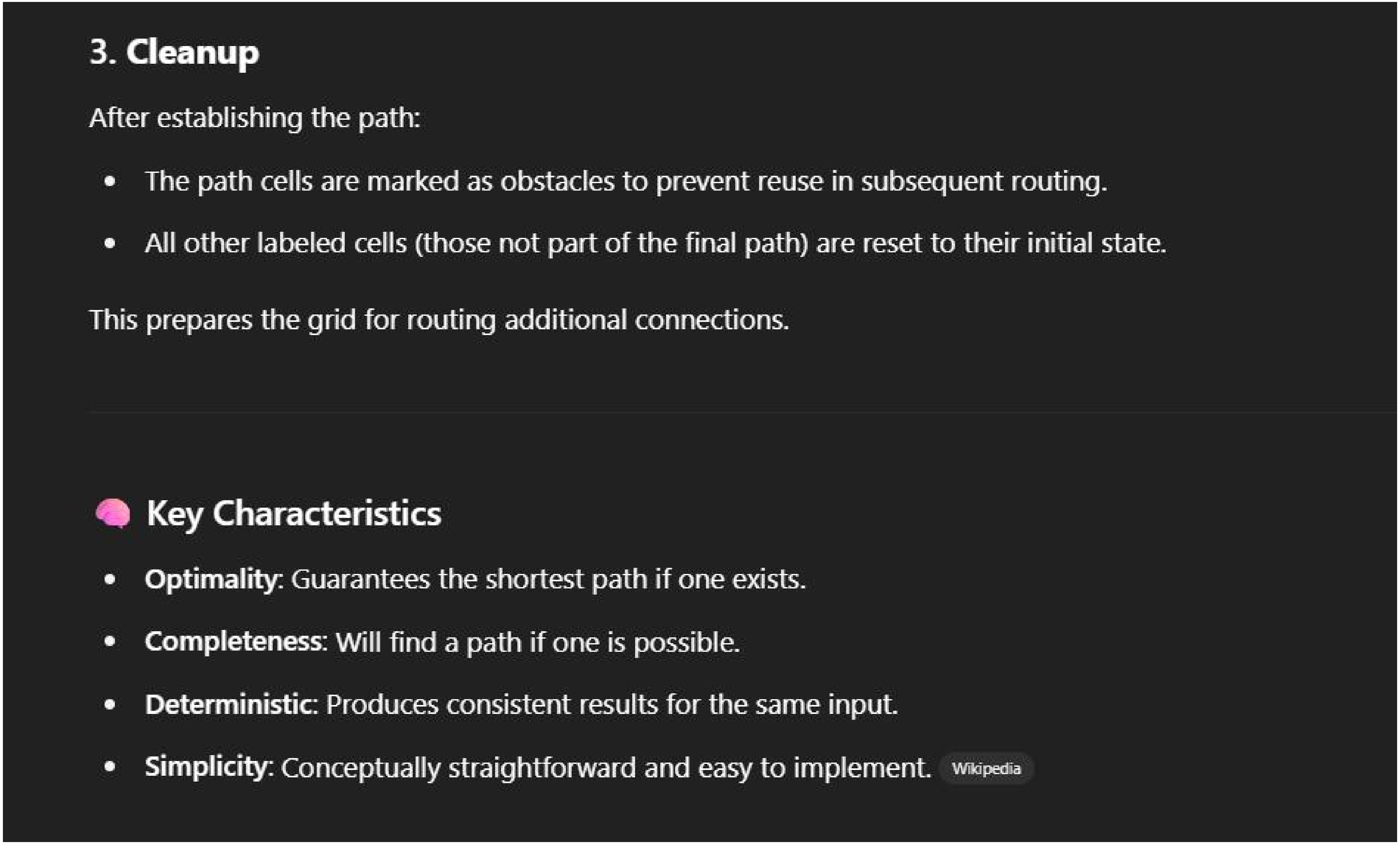
Starting from the source cell, the algorithm performs a breadth-first search (BFS), expanding outward in all four cardinal directions (up, down, left, right). Each newly reached cell is labeled with a numerical value representing its distance from the source. This process continues until the target cell is reached or all possibilities are exhausted.

This phase ensures that the shortest path, in terms of the number of steps, is found if it exists. GitHub

2. Backtracing (Path Construction Phase)

Once the target is reached, the algorithm traces back from the target to the source by moving to neighboring cells with progressively decreasing labels. This backtracing constructs the shortest path found during the wave propagation phase.

In cases where multiple neighbors have the same label, heuristics can be applied to choose the path that minimizes turns or meets other design criteria.



3. Cleanup

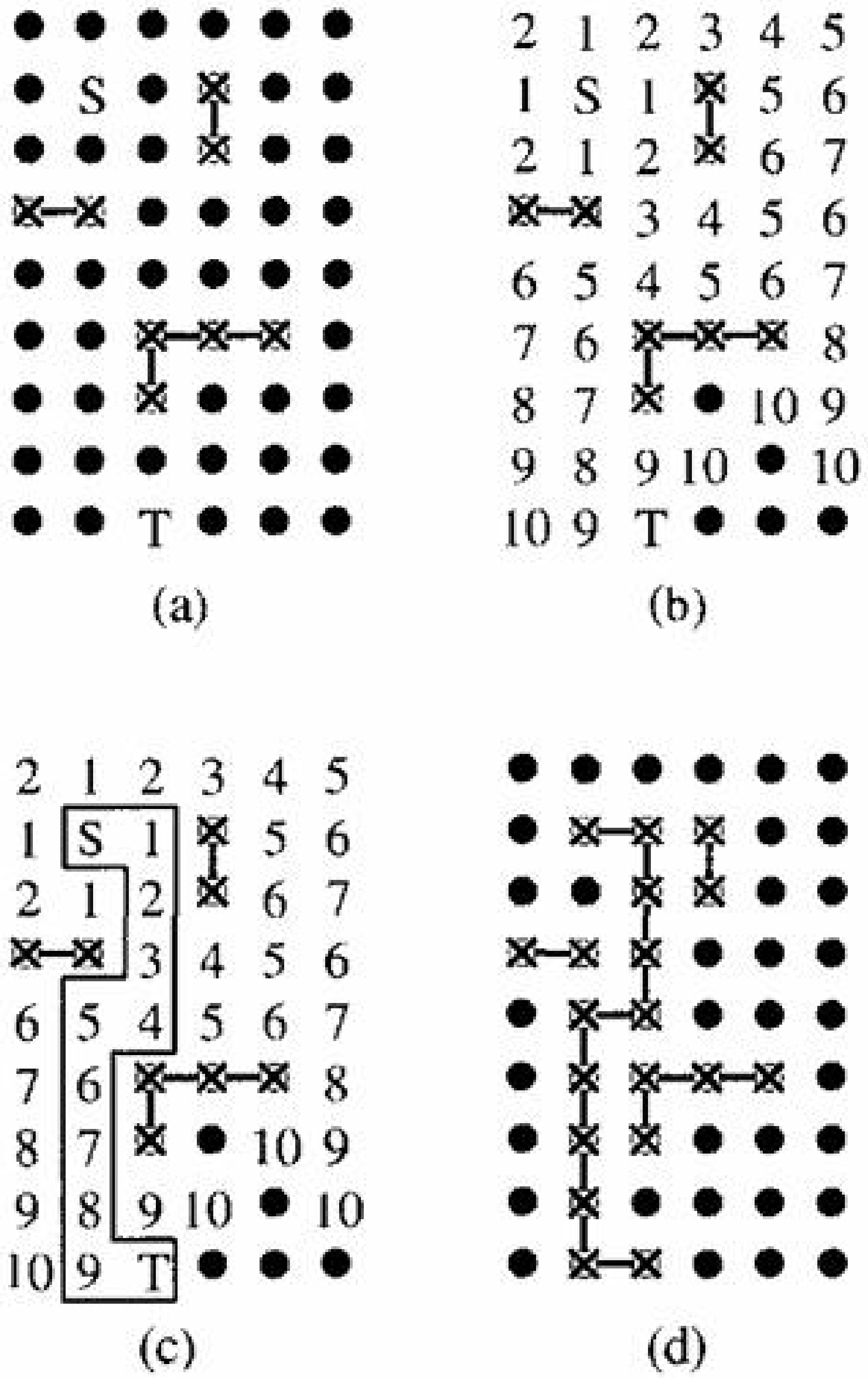
After establishing the path:

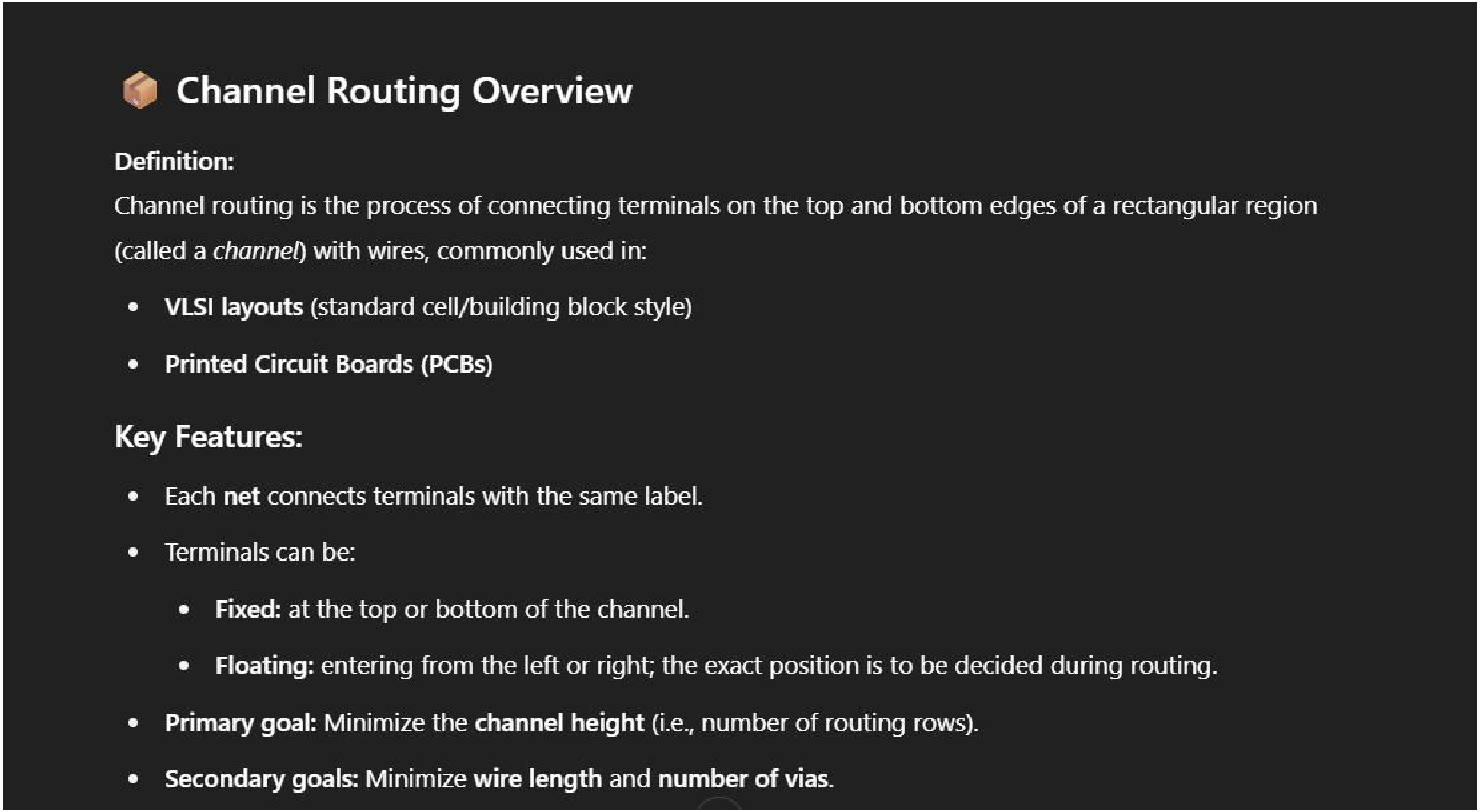
- The path cells are marked as obstacles to prevent reuse in subsequent routing.
- All other labeled cells (those not part of the final path) are reset to their initial state.

This prepares the grid for routing additional connections.

🧠 Key Characteristics

- **Optimality**: Guarantees the shortest path if one exists.
- **Completeness**: Will find a path if one is possible.
- **Deterministic**: Produces consistent results for the same input.
- **Simplicity**: Conceptually straightforward and easy to implement. Wikipedia


S
T
(a)
(b)
(c)
(d)



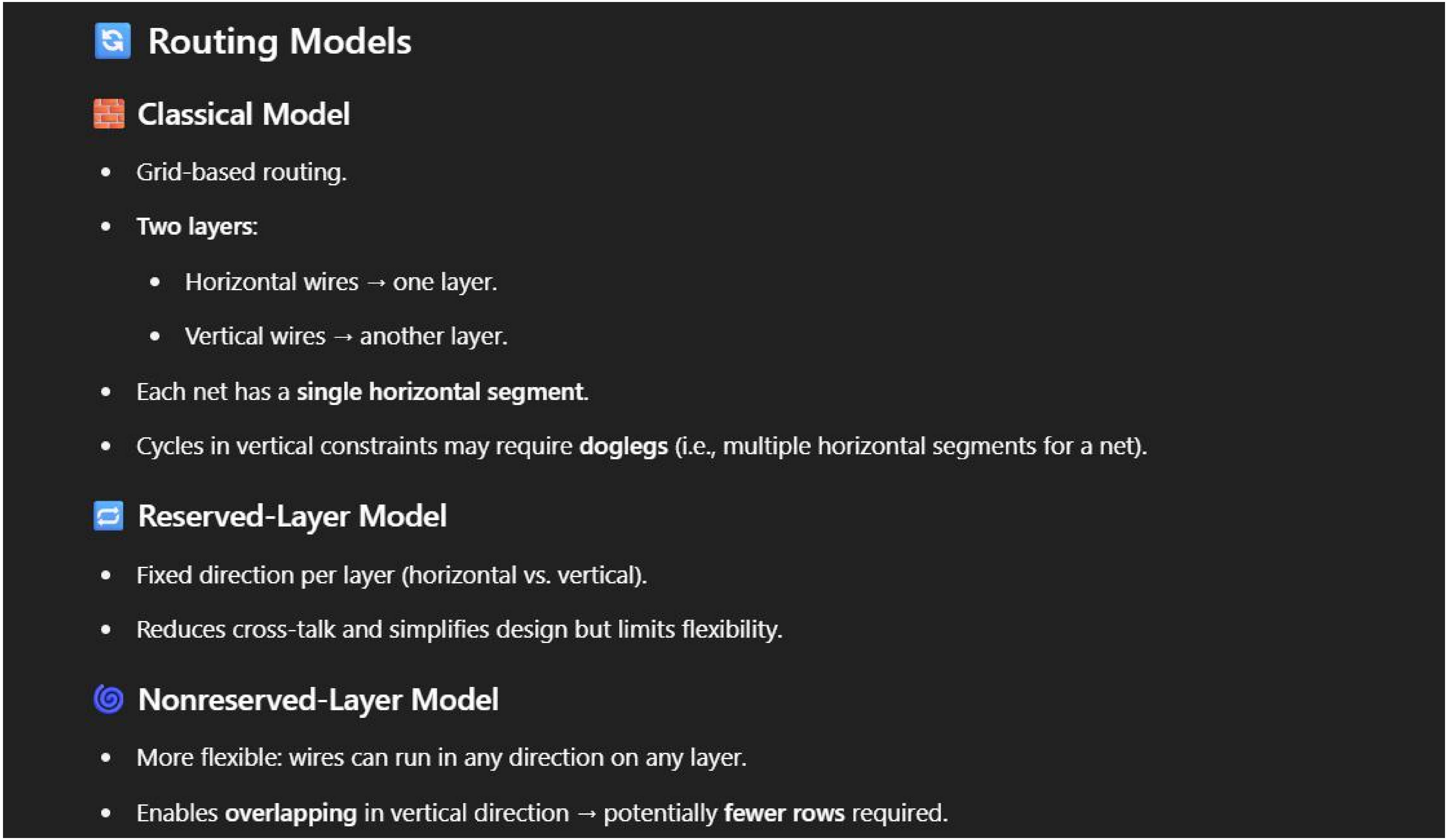
📦 Channel Routing Overview

Definition:
Channel routing is the process of connecting terminals on the top and bottom edges of a rectangular region (called a *channel*) with wires, commonly used in:

- **VLSI layouts** (standard cell/building block style)
- **Printed Circuit Boards (PCBs)**

Key Features:

- Each **net** connects terminals with the same label.
- Terminals can be:
  - **Fixed:** at the top or bottom of the channel.
  - **Floating:** entering from the left or right; the exact position is to be decided during routing.
- **Primary goal:** Minimize the **channel height** (i.e., number of routing rows).
- **Secondary goals:** Minimize **wire length** and **number of vias**.



🔄 Routing Models

🧱 Classical Model

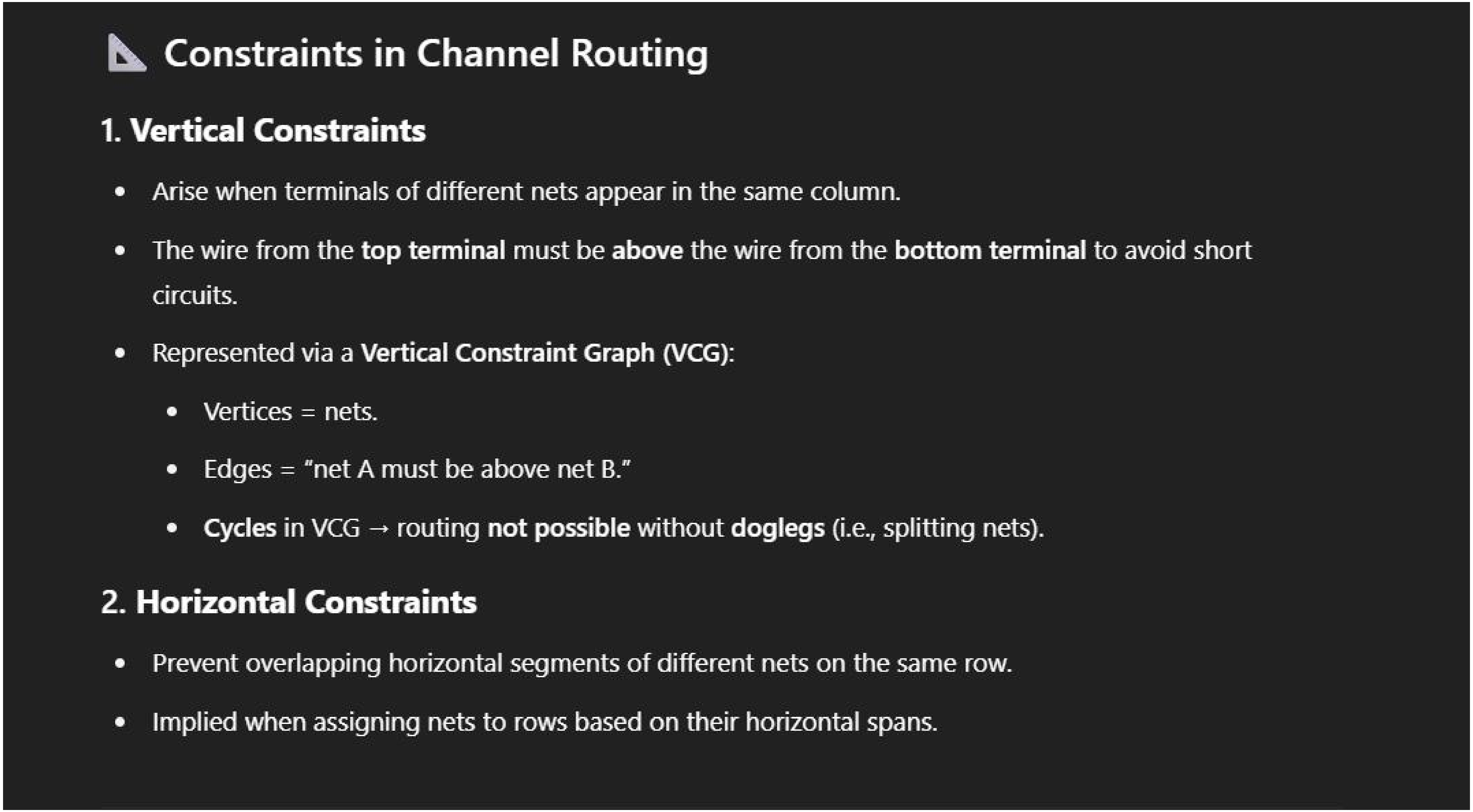
- Grid-based routing.
- **Two layers:**
  - Horizontal wires → one layer.
  - Vertical wires → another layer.
- Each net has a **single horizontal segment**.
- Cycles in vertical constraints may require **doglegs** (i.e., multiple horizontal segments for a net).

🔁 Reserved-Layer Model

- Fixed direction per layer (horizontal vs. vertical).
- Reduces cross-talk and simplifies design but limits flexibility.

🌀 Nonreserved-Layer Model

- More flexible: wires can run in any direction on any layer.
- Enables **overlapping** in vertical direction → potentially **fewer rows** required.



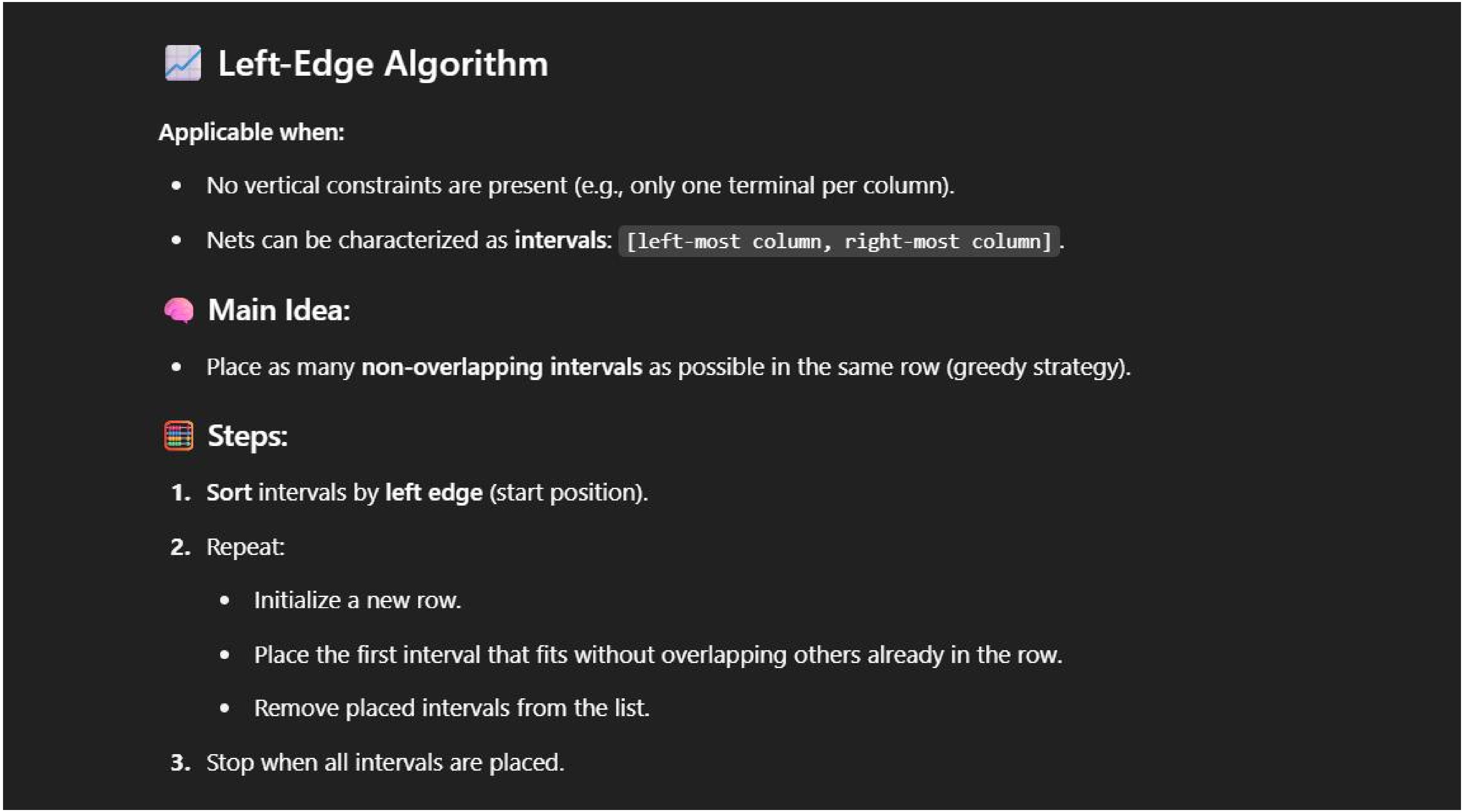
📐 Constraints in Channel Routing

1. Vertical Constraints

- Arise when terminals of different nets appear in the same column.
- The wire from the **top terminal** must be **above** the wire from the **bottom terminal** to avoid short circuits.
- Represented via a **Vertical Constraint Graph (VCG)**:
  - Vertices = nets.
  - Edges = “net A must be above net B.”
  - **Cycles** in VCG → routing **not possible** without **doglegs** (i.e., splitting nets).

2. Horizontal Constraints

- Prevent overlapping horizontal segments of different nets on the same row.
- Implied when assigning nets to rows based on their horizontal spans.



📈 Left-Edge Algorithm

Applicable when:

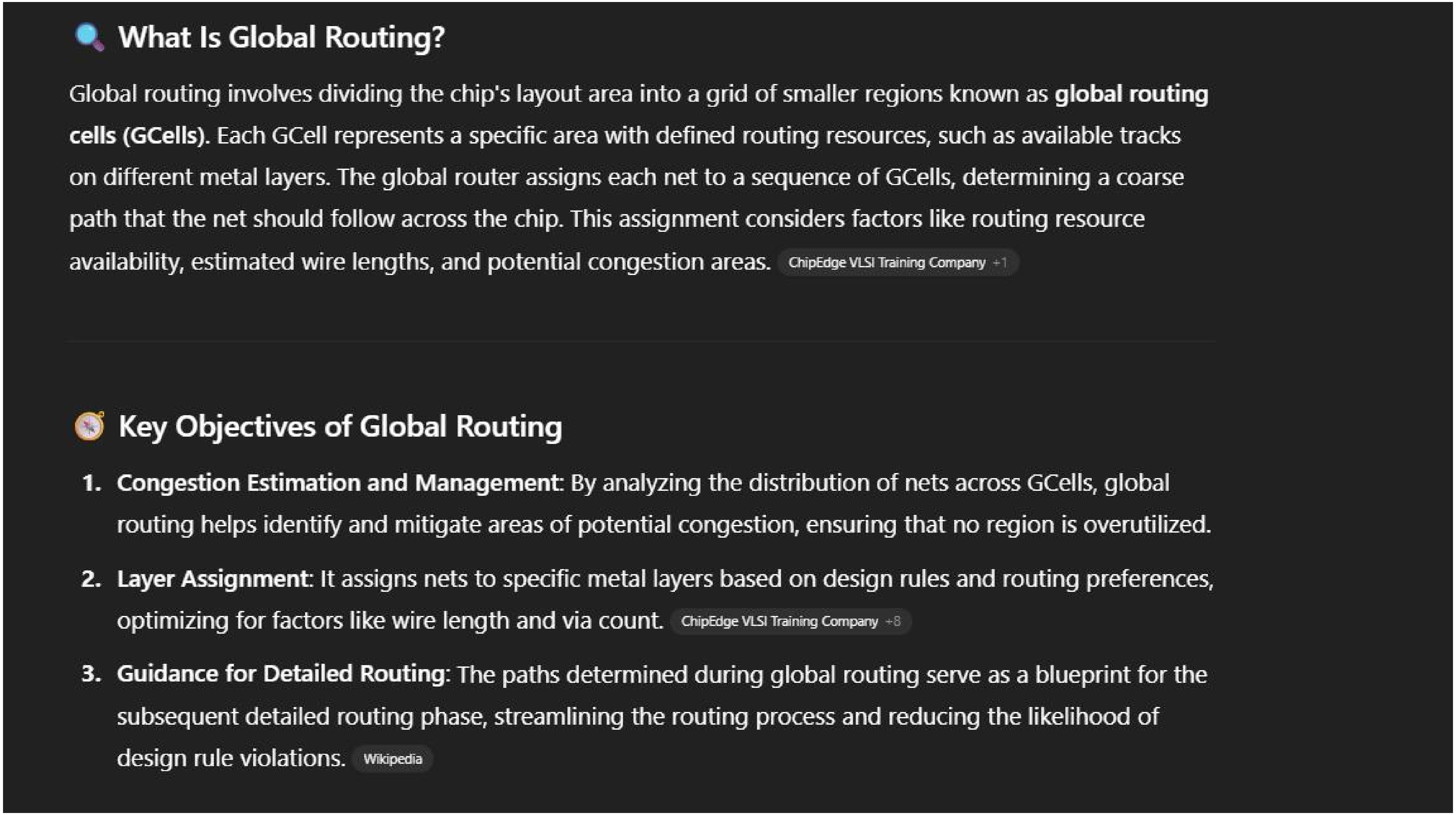
- No vertical constraints are present (e.g., only one terminal per column).
- Nets can be characterized as **intervals**: `[left-most column, right-most column]`.

🧠 Main Idea:

- Place as many **non-overlapping intervals** as possible in the same row (greedy strategy).

🧮 Steps:

1. **Sort** intervals by **left edge** (start position).
2. Repeat:
   - Initialize a new row.
   - Place the first interval that fits without overlapping others already in the row.
   - Remove placed intervals from the list.
3. Stop when all intervals are placed.

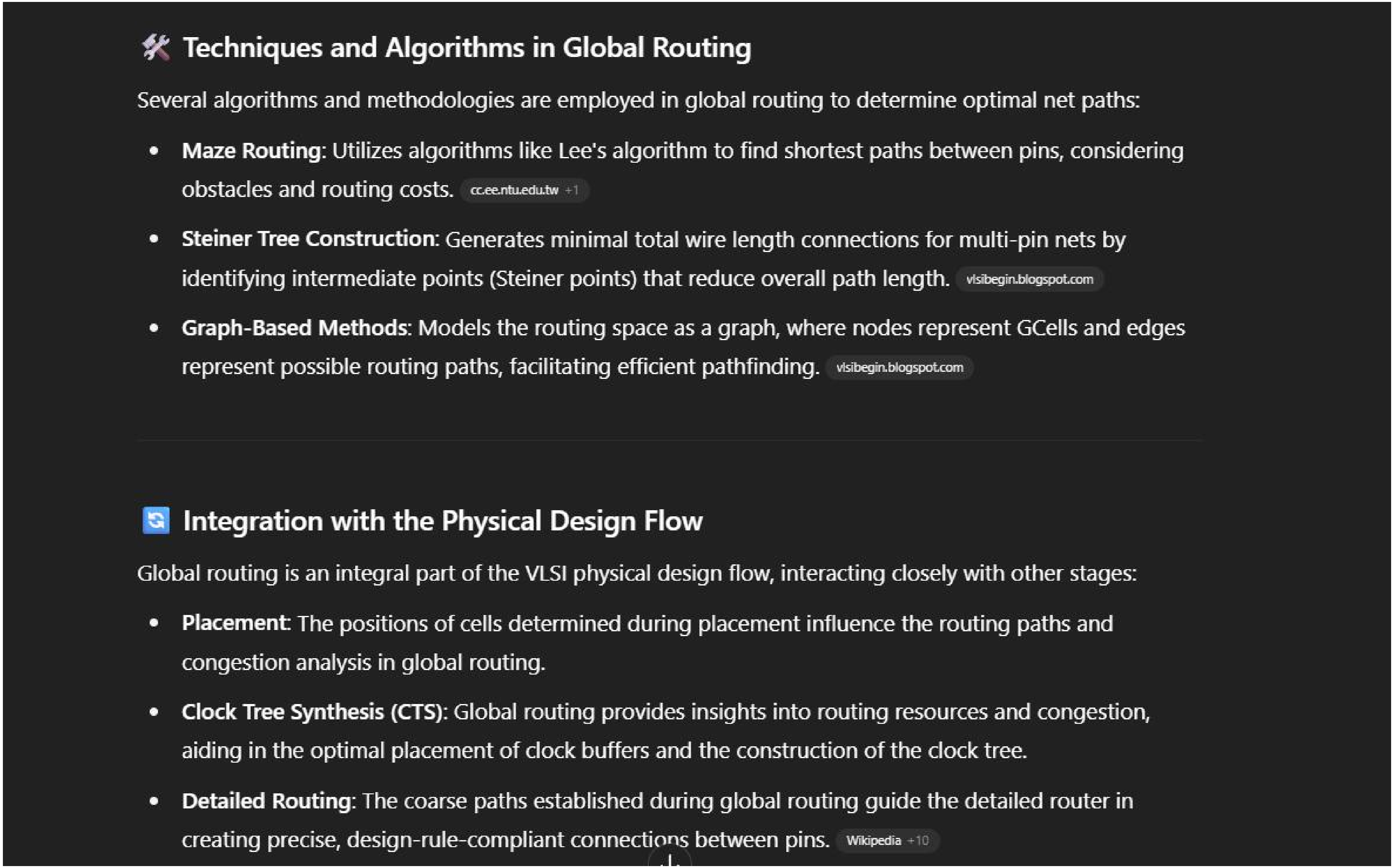


🔍 What Is Global Routing?

Global routing involves dividing the chip's layout area into a grid of smaller regions known as **global routing cells (GCells).** Each GCell represents a specific area with defined routing resources, such as available tracks on different metal layers. The global router assigns each net to a sequence of GCells, determining a coarse path that the net should follow across the chip. This assignment considers factors like routing resource availability, estimated wire lengths, and potential congestion areas. ChipEdge VLSI Training Company +1

🧭 Key Objectives of Global Routing

1. **Congestion Estimation and Management**: By analyzing the distribution of nets across GCells, global routing helps identify and mitigate areas of potential congestion, ensuring that no region is overutilized.
2. **Layer Assignment**: It assigns nets to specific metal layers based on design rules and routing preferences, optimizing for factors like wire length and via count. ChipEdge VLSI Training Company +8
3. **Guidance for Detailed Routing**: The paths determined during global routing serve as a blueprint for the subsequent detailed routing phase, streamlining the routing process and reducing the likelihood of design rule violations. Wikipedia



🛠️ Techniques and Algorithms in Global Routing

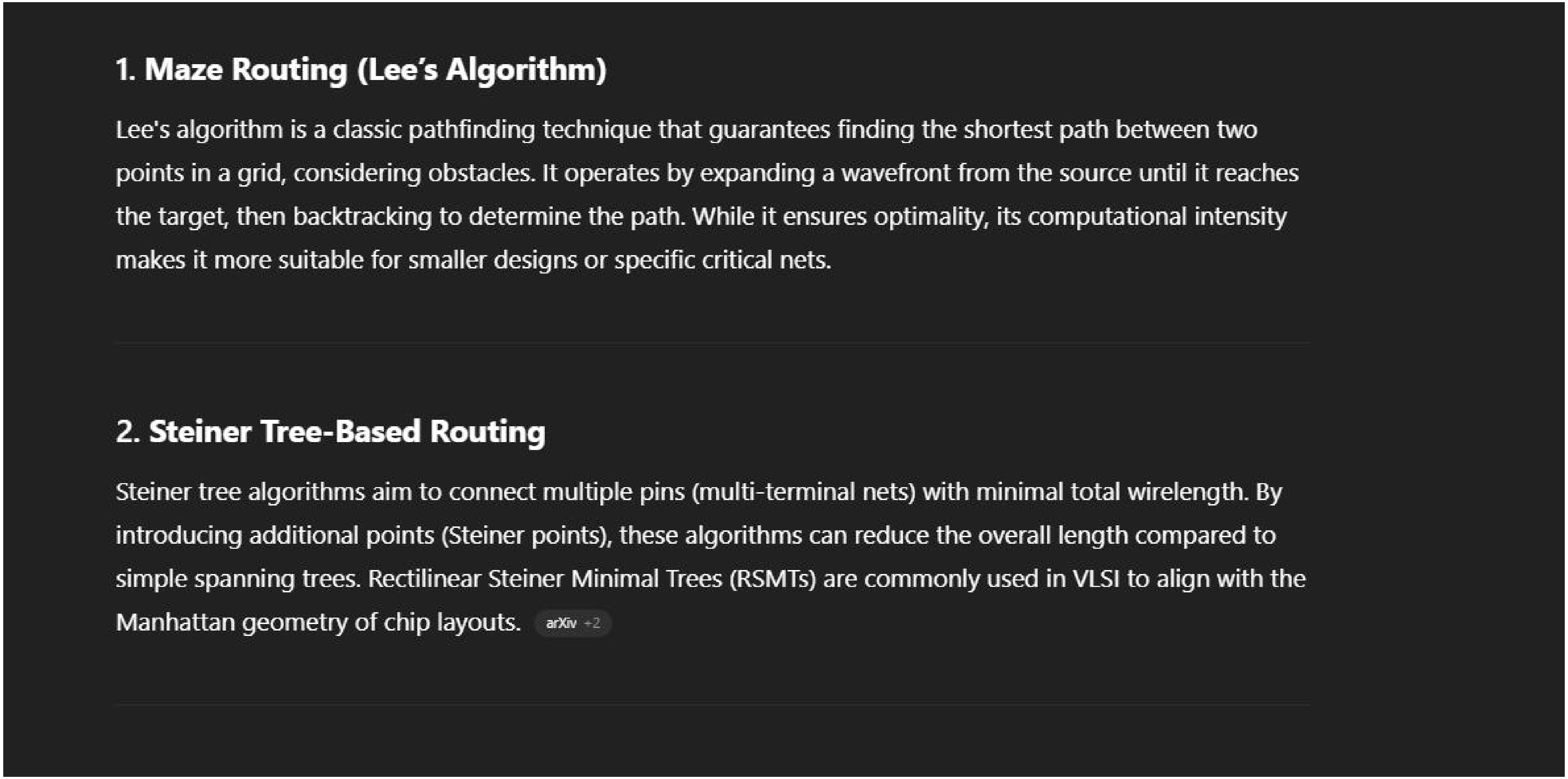
Several algorithms and methodologies are employed in global routing to determine optimal net paths:

- **Maze Routing**: Utilizes algorithms like Lee's algorithm to find shortest paths between pins, considering obstacles and routing costs. cc.ee.ntu.edu.tw +1
- **Steiner Tree Construction**: Generates minimal total wire length connections for multi-pin nets by identifying intermediate points (Steiner points) that reduce overall path length. vlsibegin.blogspot.com
- **Graph-Based Methods**: Models the routing space as a graph, where nodes represent GCells and edges represent possible routing paths, facilitating efficient pathfinding. vlsibegin.blogspot.com

🔄 Integration with the Physical Design Flow

Global routing is an integral part of the VLSI physical design flow, interacting closely with other stages:

- **Placement**: The positions of cells determined during placement influence the routing paths and congestion analysis in global routing.
- **Clock Tree Synthesis (CTS)**: Global routing provides insights into routing resources and congestion, aiding in the optimal placement of clock buffers and the construction of the clock tree.
- **Detailed Routing**: The coarse paths established during global routing guide the detailed router in creating precise, design-rule-compliant connections between pins. Wikipedia +10

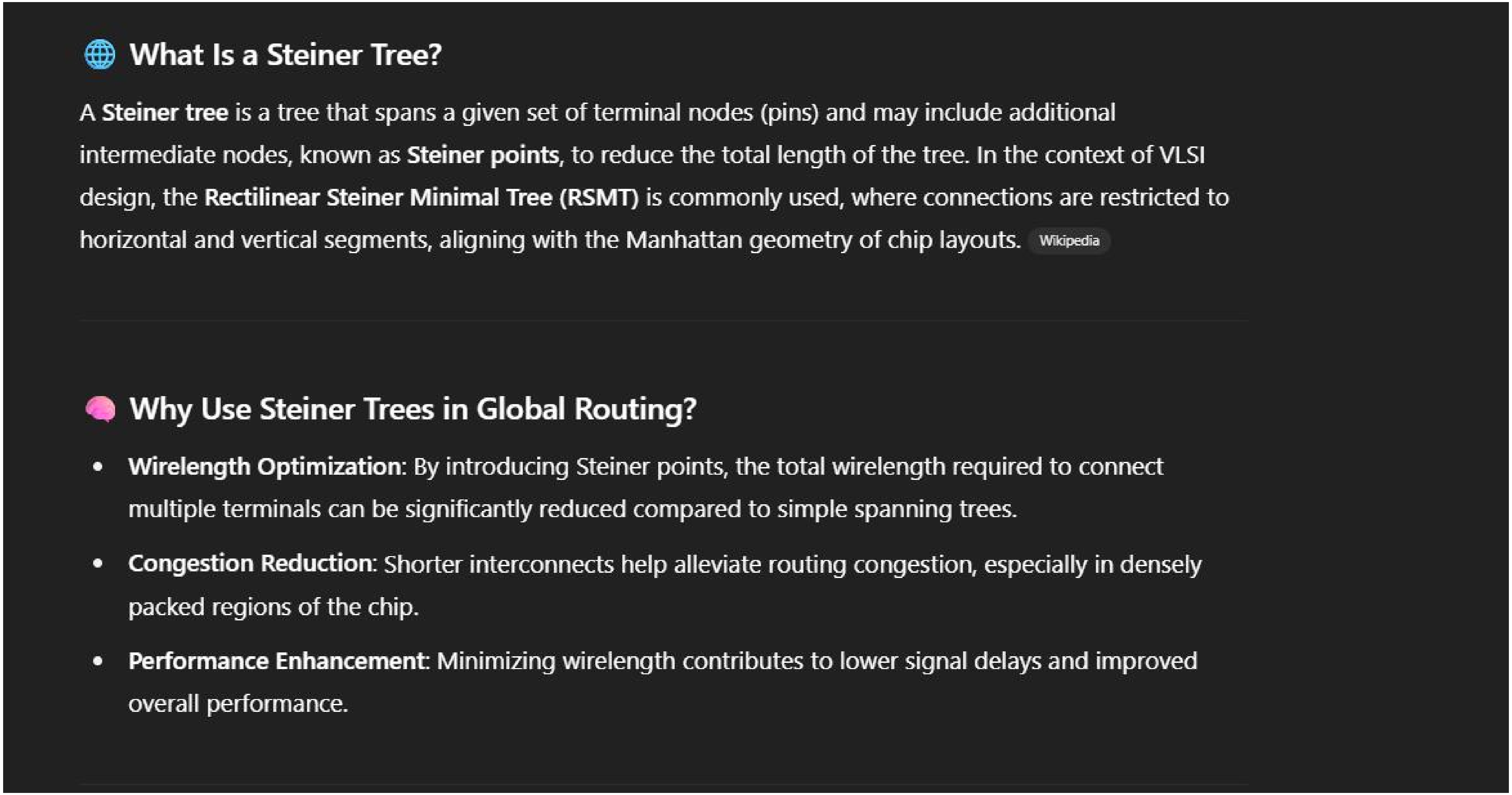


1. Maze Routing (Lee’s Algorithm)

Lee's algorithm is a classic pathfinding technique that guarantees finding the shortest path between two points in a grid, considering obstacles. It operates by expanding a wavefront from the source until it reaches the target, then backtracking to determine the path. While it ensures optimality, its computational intensity makes it more suitable for smaller designs or specific critical nets.

2. Steiner Tree-Based Routing

Steiner tree algorithms aim to connect multiple pins (multi-terminal nets) with minimal total wirelength. By introducing additional points (Steiner points), these algorithms can reduce the overall length compared to simple spanning trees. Rectilinear Steiner Minimal Trees (RSMTs) are commonly used in VLSI to align with the Manhattan geometry of chip layouts. arXiv +2



🌐 What Is a Steiner Tree?

A **Steiner tree** is a tree that spans a given set of terminal nodes (pins) and may include additional intermediate nodes, known as **Steiner points**, to reduce the total length of the tree. In the context of VLSI design, the **Rectilinear Steiner Minimal Tree (RSMT)** is commonly used, where connections are restricted to horizontal and vertical segments, aligning with the Manhattan geometry of chip layouts. Wikipedia

🧠 Why Use Steiner Trees in Global Routing?

- **Wirelength Optimization:** By introducing Steiner points, the total wirelength required to connect multiple terminals can be significantly reduced compared to simple spanning trees.
- **Congestion Reduction:** Shorter interconnects help alleviate routing congestion, especially in densely packed regions of the chip.
- **Performance Enhancement:** Minimizing wirelength contributes to lower signal delays and improved overall performance.