論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Routing Strategies for Critical Routing Layers in Modern Advanced Chips
著者(和文)	WANG Zezhong
Author(English)	WANG Zezhong
出典(和文)	学位:博士(学術), 学位授与機関:東京科学大学, 報告番号:甲第381号, 授与年月日:2025年3月26日, 学位の種別:課程博士, 審査員:髙橋 篤司,一色 剛,佐々木 広,藤木 大地,ISLAM A K M MAHFUZUL
Citation(English)	Degree:Doctor (Academic), Conferring organization: Institute of Science Tokyo, Report number:甲第381号, Conferred date:2025/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Routing Strategies for Critical Routing Layers in Modern Advanced Chips

WANG Zezhong

Department of Information and Communications Engineering

Institute of SCIENCE TOKYO

March 2025

Abstract

Modern advanced chips, such as 3D flash memory, feature numerous regularly placed bonding pads used for communication between a CMOS chip and a flash memory chip. These pads act as obstacles during routing, resulting in constrained horizontal capacity within these layers. A routing layer with limited horizontal capacity is called a critical routing layer. This research models a critical routing layer as a generalized channel. To address limited horizontal capacity, a versatile routing framework is introduced to accomplish various routing objectives while minimizing track usage. The SDG algorithm is designed to enhance chip performance by reducing the total vertical length. To alleviate local congestion, the UEO algorithm is proposed as an extension of SDG. These strategies boost VLSI performance, supporting advanced and efficient electronics.

Contents

Chapter 1

Introduction

1.1 Background

Global routing is a fundamental stage in very-large-scale integration (VLSI) physical design that determines the approximate paths of interconnections between circuit components while considering routing constraints. It partitions the chip into a grid structure and assigns nets to routing regions, ensuring that the total routing resource usage is minimized while meeting design rules. Unlike detailed routing, which specifies exact routing pattern, global routing provides a high level routing pattern that guides the subsequent stages of physical design.

The effectiveness of global routing directly influences critical chip performance metrics such as signal integrity, power consumption, and timing. Efficient global routing reduces wire length, minimizing signal delays and power dissipation. Additionally, it helps prevent routing congestion, which can lead to timing violations and increased manufacturing complexity. Poor global routing results in excessive detours, increased via usage, and local congestion, ultimately degrading chip reliability and manufacturability.

As semiconductor technology advances, global routing faces increasing challenges due to shrinking feature sizes, higher integration densities, and complex design constraints. Modern chips, such as 3D integrated circuits and advanced memory architectures, impose stricter limitations on routing resources, making congestion management

more difficult. Additionally, manufacturing variations and multi-patterning constraints further complicate the routing process. This complexity is not only reflected in the integration of more transistors within a chip but also in the routing design to obtain high-performance chips [1]. As a result, conventional global routing techniques struggle to meet the demands of emerging technologies, necessitating the development of more sophisticated routing strategies.

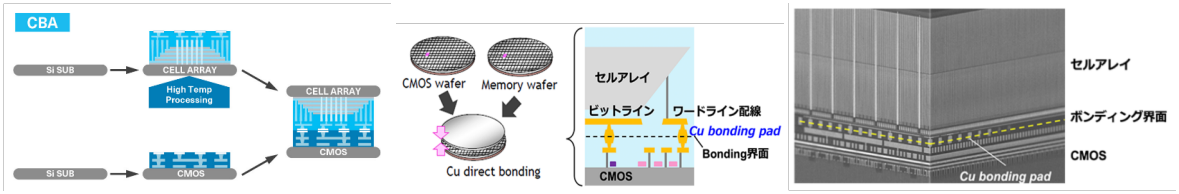
This research proposes routing strategies tailored for critical routing layers in modern advanced chips that utilize new manufacturing technologies. The proposed strategies effectively minimize track usage, enabling designers to maximize limited routing resources. Additionally, they reduce total vertical wire length and alleviate local congestion while maintaining minimal track usage. By optimizing routing efficiency, these strategies contribute to achieving broader routing objectives, enhancing the overall performance and manufacturability of advanced semiconductor designs.

1.2 Modern Advanced Chip

This section introduces CBA technology and the critical routing layer in modern advanced chips that adopt it. This research focus on modern advanced chip such as 3D flash memory. In recent years, the field of semiconductor technology has seen tremendous advancements, particularly in the area of 3-dimensional (3D) chip manufacturing. This innovative approach allows for the stacking of multiple 2D chip layers to form a 3D structure, which provides numerous benefits such as increased integration density, improved performance, and reduced power consumption. However, one of the key challenges in achieving effective 3D integration is the development of reliable interconnection technology between these vertically stacked 2D chips.

1.2.1 CBA Technology

To achieve reliable interconnection in vertically stacked 2D chips, a novel integration technology known as CMOS Directly Bonded to Array (CBA) has been introduced [2, 3, 4]. In CBA, two silicon wafers one containing the CMOS circuit and another the memory array are fabricated separately and then directly bonded together, as shown in 1.1. This approach eliminates the need for conventional through silicon vias (TSVs)



(a) CBA Flow (b) CBA Details (c) CBA SEM Image

Figure 1.1: CBA Technology[2]

or micro bumps, reducing parasitic resistance and enhancing signal integrity. Communication between the CMOS and flash memory chips is achieved through bonding pads, which serve as interconnection points. However, these bonding pads introduce additional routing constraints, as they must be treated as obstacles during the global routing process, thereby affecting routing resources allocation and increasing design complexity.

1.2.2 Critical Routing Layer

The adoption of CBA technology introduces a significant number of regularly placed bonding pads, which are treated as obstacles during the routing process. This results in reduced routing flexibility and increased congestion, particularly in layers where one direction routing resources are constrained. Additionally, due to the architectural constraints of CMOS chips and the design policies of 3D flash memory, certain routing regions within the CMOS chip inherently exhibit bottlenecks in one routing direction, further exacerbating congestion.

In this research, a routing layer with limited horizontal routing capacity is termed a cal routing layer. As shown in Fig. 1.2, the 2D top view of a routing layer within a CMOS chip illustrates how bonding pads, arranged in a regular array for 3D integration, occupy significant routing space, further restricting the available routing resources. The presence of these obstacles requires advanced routing strategies to efficiently utilize the remaining routing capacity while minimizing congestion and ensuring reliable interconnections.

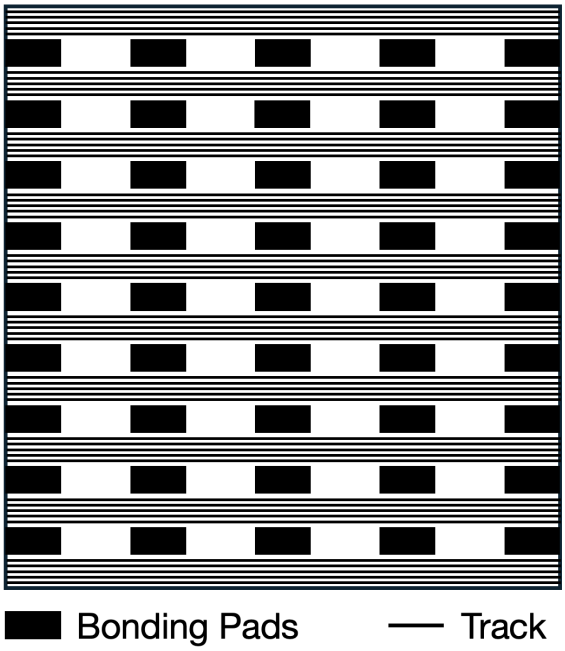


Figure 1.2: A routing layer for horizontal wires with tight capacity due to regularly placed bonding pads.

1.3 Objective

This research develops effective routing strategies for the critical routing layer in advanced chips, addressing limited horizontal routing capacity through an efficient model that captures resource constraints. Based on this model, a versatile routing framework is introduced to minimize track usage and optimize resource utilization. To further improve efficiency, a length minimization algorithm reduces global routing congestion, while a local congestion alleviation algorithm enhances routing feasibility and increases the completion rate in detailed routing.

In Chapter 1, this research highlights the growing importance of 3D integration technology in modern semiconductor devices. Specifically, we discussed the use of CBA technology, which are crucial in the development of advanced products such as CMOS image sensors and 3D flash memory. However, these technologies introduce new challenges, especially in routing design, where a huge number of bonding pads, arranged regularly, obstruct traditional automated routing methods. This research introduces innovative routing strategies to address these issues and improve routing results in 3D IC implementations.

Chapter 2 provides an overview of related research and introduces the *Generalized Channel Routing Problem* (GCRP). In conventional two-layer standard cell routing, pins are placed along channel boundaries. In contrast, in generalized channels of 3D integrated circuits, pins can also be inside the channel, while horizontal routing resources are limited. These constraints challenge conventional automatic routing techniques. To achieve efficient, high-quality routing, this research emphasizes a generalized single-trunk channel routing method, where each net is routed using a single-trunk Steiner tree.

Chapter 3 introduces the *Greedy Routing Framework with Guarantee* (GRFG), a framework designed to address GCRP. GRFG sequentially assigns routing trunks, ensuring no overlap while minimizing track usage. It guarantees routing completion for each net and allows flexible net prioritization based on design objectives, enabling diverse routing solutions tailored to various requirements.

Chapter 4 introduces the *Symmetric Difference General* (SDG) channel routing algorithm, which minimizes total vertical wire length in GCRP. By assigning nets to tracks based on symmetric difference priority, SDG effectively reduces vertical wire length. Built upon GRFG, this heuristic algorithm achieves length minimization while ensuring minimized track usage.

Chapter 5 introduces the *Under, Enclose, Over* (UEO) channel routing algorithm, which addresses local congestion in GCRP. UEO reduces local congestion by defining the range of tracks where net assignment could increase congestion and by avoiding these tracks when assigning nets. While this results in a slight increase in total vertical wire length compared to SDG, UEO significantly alleviates local congestion, ensuring more efficient routing in congested areas.

Chapter 6 summarizes the key achievements of this research, highlighting the contributions of the proposed routing strategies to 3D integrated circuit design. The chapter discusses their role in improving routing efficiency for critical layers and explores potential enhancements by integrating real world design constraints.

Chapter 2

Channel Routing

The critical routing layer in modern advanced chips shares key characteristics with the channel routing model, making it a suitable abstraction for addressing routing challenges. In CBA technology, the high density of regularly placed bonding pads severely constrains routing resources, particularly in one direction. This research assumes limited horizontal routing capacity, creating a structured scenario where nets must be assigned to tracks within a confined region, similar to a channel. In both cases, net trunks are routed along predefined horizontal tracks, while vertical wires occupy other metal layers. Modeling the critical routing layer as a generalized channel enables the application of established channel routing techniques to minimize track usage, alleviate congestion, and enhance routing efficiency in modern chip designs.

The concept of VLSI channel routing was first introduced by Hashimoto and Stevens in their work [5]. In this context, a channel refers to a rectangular area designated for routing, bounded by two rows of pins positioned along the top and bottom edges. Connections between pins are established using line segments that follow orthogonal grids, known as horizontal and vertical tracks. These tracks are kept separate, with connections between them facilitated by holes located at the grid intersections. A net is defined as a set of pins that need to be connected. Within the channel, pins belonging to the same net are connected, while those of different nets remain isolated from each other. The primary goal in channel routing is to minimize the channel height. The number of horizontal tracks required to connect pins of each net. Channel density is another critical factor, defined as the maximum number of nets intersected by a vertical line drawn through the channel. A net is considered intersected by a vertical

line if it has pins on both sides of the line or if a pin lies directly on the line. The channel density provides a basic lower bound for the channel height.

2.1 Classical Channel Routing Problem (CCRP)

The *Classical Channel Routing Problem* (CCRP) addresses routing scenarios where pins are strictly located along the upper and lower boundaries of a designated channel. Numerous algorithms have been developed to optimize this problem, forming the foundation of early VLSI routing techniques. Hashimoto and Stevens [5] pioneered channel assignment optimization, while Deutsch [6] introduced the dogleg channel router, a method allowing segment reallocation for improved flexibility. LaPaugh [7] analyzed the complexity of circuit layouts through an analytical framework, providing deeper insights into routing constraints. Yoshimura and Kuh [8] proposed efficient algorithms that refined channel routing strategies. Szymanski [9] proved the NP-completeness of dogleg channel routing, underscoring the problem's computational difficulty. Ho et al. [10] introduced a general greedy algorithm to enhance channel routing efficiency under various constraints. Takahashi and Murata [11] extended routing strategies by developing a three-layer L-shaped channel routing algorithm to accommodate complex layouts. Sau et al. [12] later introduced a graph-based algorithm focused on minimizing total wire length. Recent advancements in CCRP continue to refine its methodologies. Taniguchi et al. [13, 14] explored bottleneck channel routing techniques to optimize area utilization in analog VLSI designs. Further research by Taniguchi's team [15, 16] introduced algorithms that leverage track sharing across adjacent layers, effectively improving routing feasibility in high-density regions.

Despite extensive research and algorithmic advancements, CCRP remains inadequate for routing layers with a high density of obstacles, such as those in advanced chip designs. Its primary limitation is the assumption that all pins are fixed along the channel boundaries, making it unsuitable for complex routing layers with obstacles and irregularly placed terminals. Since classical channel models routing as a localized problem within predefined boundaries, applying it to critical routing layers necessitates segmenting the design into multiple independent channels, leading to fragmented solutions and increased computational overhead, as shown in Fig. 2.1. This segmentation not only prolongs routing time but also introduces inefficiencies, including increased wire length and suboptimal track utilization. Furthermore, CCRP lacks the flexibil-

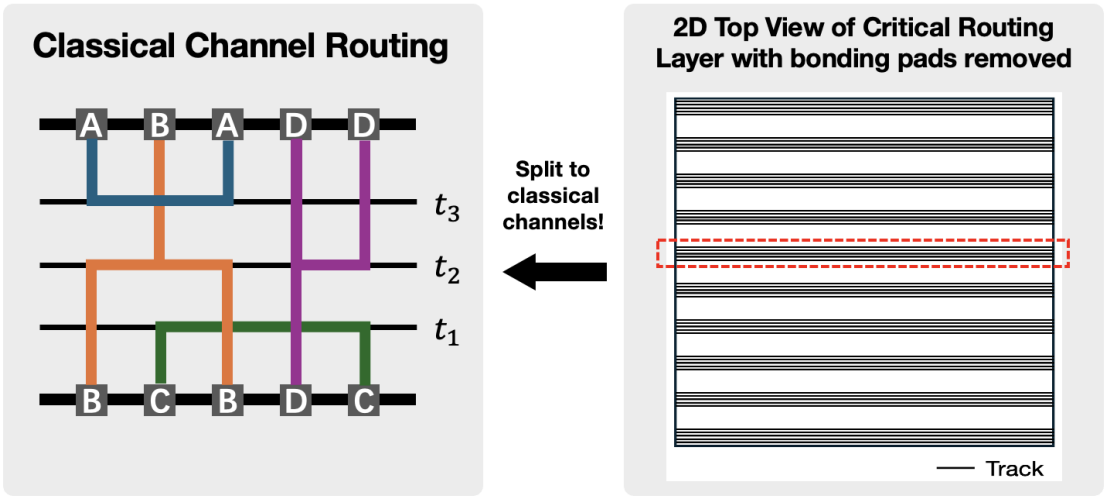


Figure 2.1: CCRP

ity to adapt dynamically to varying pin distributions and obstacle locations, making it unsuitable for critical layers where resource allocation must be globally optimized. Consequently, alternative routing models that generalize channel routing to accommodate real-world constraints are essential for addressing modern chip design challenges.

2.2 Generalized Channel Routing Problem (GCRP)

To optimize routing on critical layers with constrained horizontal capacity, this research models such layers as *generalized channels*. Unlike CCRP, which divides the routing area into multiple independent regions, the generalized channel provides a unified routing space where predefined horizontal tracks span the entire area while avoiding bonding pads, which serve as routing obstacles. This approach preserves global horizontal routing capacity and eliminates the need for explicit segmentation into smaller channels, thereby improving routing efficiency in layers with a high density of regularly placed obstacles, as shown in Fig. 2.2.

While the generalized channel resembles the classical channel model used in standard cell designs, it differs by supporting more complex routing constraints, making it particularly suitable for modern integrated circuits with dense component placement. The ability to handle obstacles efficiently without fragmenting the problem into multiple sub-problems allows for greater flexibility in routing solutions, reducing unnecessary

detours and optimizing wire length.

The routing challenges within this model constitute the *generalized channel routing problem (GCRP)* [17], which extends CCRP formulations to handle highly constrained environments. This problem requires careful track assignment to ensure efficient wire utilization while mitigating congestion and ensuring minimal interference between nets.

A fundamental aspect of VLSI routing is the interconnection of multiple pins in a net, typically addressed using Steiner trees. In an unrestricted Euclidean plane, the Steiner minimal tree (SMT) provides the shortest interconnection between pins. However, due to the grid-based nature of VLSI layouts, the rectilinear Steiner minimal tree (RSMT) is preferred [18, 19, 20]. While RSMT minimizes wire length in a rectilinear space, it can lead to excessive horizontal wire usage, making it inefficient for critical routing layers with limited horizontal capacity. This overuse of horizontal tracks can cause congestion, reducing the feasibility of efficient solutions in dense layouts.

To alleviate horizontal congestion, dogleg routing [6] introduces additional vertical segments by segmenting a trunk into several sections, facilitating more efficient distribution of nets across available tracks. This technique, commonly used in CCRP, optimizes track utilization under vertical constraints. However, while dogleg routing reduces horizontal congestion, it increases the number of vias and vertical wires. Excessive via usage can degrade signal integrity, increase resistance and delay, and present additional challenges during later stages of detailed routing.

An alternative approach, the single-trunk Steiner tree (STST), minimizes horizontal routing resource usage. By maintaining a single trunk per net and minimizing vertical connections, STST reduces both horizontal track consumption and via insertion. This approach is particularly beneficial in routing environments with tight horizontal capacity. Unlike RSMT, which may introduce multiple horizontal branches for a single net, STST confines horizontal usage to a single track, ensuring better resource allocation while simplifying subsequent routing stages.

The choice of routing strategy within GCRP has a significant impact on total wire length, congestion. While conventional methods like RSMT and dogleg routing offer certain benefits, their limitations are more apparent in dense routing environments where horizontal resources are severely constrained. STST offers a viable alternative by minimizing track consumption and via usage, making it an attractive option for advanced VLSI designs with stringent constraints.

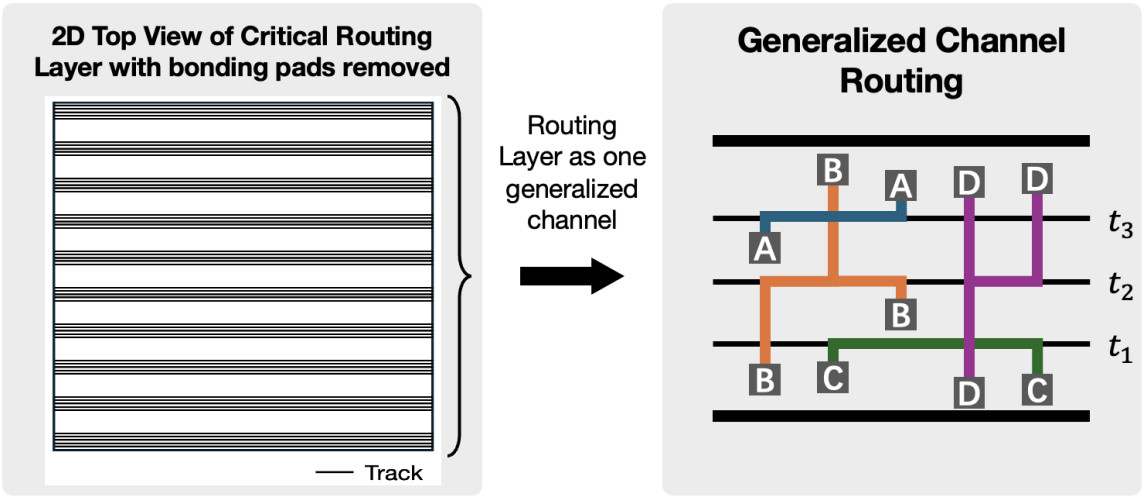


Figure 2.2: GCRP

2.3 Routing Constraints

In VLSI routing, constraints define the allowed routing paths and influence the overall efficiency, manufacturability and performance of the circuit. Among these, *horizontal constraints* and *vertical constraints* are two fundamental factors that directly impact the routing pattern of a chip. Properly handling these constraints ensures that routing resources are optimally utilized while avoiding design rule violations and excessive congestion.

2.3.1 Horizontal Constraint

A *horizontal constraint* occurs when two or more nets have horizontal segments that would overlap if placed on the same track. In GCRP, where horizontal routing is confined to predefined tracks, these constraints govern the assignment of nets to available tracks, preventing conflicts. The main objective in addressing horizontal constraints is to assign each net's trunk to a unique track or use alternative strategies, such as dogleg routing, to segment connections while preserving design feasibility.

Horizontal constraints are crucial for routing optimization, as violations can lead to short circuits or design rule violations. Track assignment algorithms and heuristic

based optimizations are commonly employed to efficiently distribute horizontal segments across available tracks while minimizing wire length and congestion.

2.3.2 Vertical Constraint

A *vertical constraint* exists when two nets have pins located in the same column, requiring careful management of vertical wire segments. Unlike horizontal routing, which is restricted to predefined tracks, vertical routing typically spans multiple metal layers, offering greater flexibility in resolving conflicts. However, excessive vertical wire usage can lead to increased via counts, higher resistance, and potential signal integrity issues.

In the context of GCRP, we assume that *vertical constraints do not exist*. This assumption is valid given the nature of GCRP, where the input netlist typically consists of subnets, and pin locations are defined to allow flexibility. Specifically:

- Pins of nets are defined virtually in global routing, meaning their locations in the input are not absolute but correspond to module placements or adjacent layers.
- Vertical wire segments required for interconnection are assumed to be implemented on other metal layers, independent of the horizontal track assignment.
- If conflicts arise from vertical wire overlaps, they can be resolved through adjustments in module design, placement flexibility, or modifications in the detailed routing stage.

Since modern VLSI design methodologies allow flexibility in module placement and inter-layer connectivity, it is reasonable to assume that pin positions are adjustable enough to eliminate vertical constraints while still meeting strict horizontal connection requirements. However, this flexibility is not unlimited. While minor adjustments can prevent vertical conflicts, horizontal connectivity demands remain rigid due to track-based routing constraints.

2.4 Summary

This chapter explored the evolution of channel routing methodologies, beginning with the *Classical Channel Routing Problem* (CCRP), which restricts routing within predefined channel boundaries. While CCRP has served as the foundation for early VLSI routing techniques, its limitations, particularly its inability to handle obstacles and irregular pin distributions, make it inadequate for modern routing layers with complex constraints. To address these challenges, the *Generalized Channel Routing Problem* (GCRP) extends the classical channle model by introducing a unified routing space where horizontal tracks are predefined, ensuring efficient resource utilization while reducing fragmentation.

A key aspect of GCRP is its ability to minimize horizontal track usage through effective track assignment strategies. Unlike the Rectilinear Steiner Minimum Tree (RSMT) and dogleg routing, which are not well-suited for GCRP due to their excessive horizontal wire usage and reliance on vertical vias, the Single Trunk Steiner Tree (STST) is a more efficient structure. STST minimizes available track resources by assigning each net to a single trunk, ensuring optimal routing within the constraints of GCRP.

Additionally, routing in both CCRP and GCRP is governed by horizontal and vertical constraints. Horizontal constraints dictate track assignments and are critical to ensuring design rule compliance. Vertical constraints, while typically managed through multilayer routing, are assumed to be ignored in GCRP due to the flexibility in pin placements and inter-layer connectivity.

Overall, this chapter establishes the theoretical framework for channel routing, highlighting the fundamental challenges and strategies necessary for modern VLSI designs. The insights presented here provide the foundation for subsequent discussions on optimization techniques and advanced routing models tailored for contemporary semiconductor technologies.

Chapter 3

Routing Framework of GCRP

In GCRP, where routing capacity and tracks are predefined, minimizing track usage is crucial to addressing tight horizontal capacity constraints. This is achieved by maximizing net assignments to tracks. This research models GCRP as an assignment problem, where trunks of each net are assigned to tracks. To solve this, a Greedy Routing Framework with Guarantee (GRFG) is proposed for sequential trunk assignment. The framework ensures successful assignment as long as the number of available tracks meets or exceeds the maximum congestion of the trunk, with the sole constraint being that trunks must not overlap. This chapter presents the GCRP formulations, critical zones, the details of GRFG, and a supporting theorem.

3.1 Formulations of GCRP

GCRP is governed by three key parameters: *density*, *capacity*, and *slack*. These parameters are critical in determining the feasibility of routing solutions and the efficiency of track assignments. *Density* represents the number of nets passing through a given coordinate, while *capacity* defines the total number of available tracks. *Slack* measures the difference between capacity and density, indicating the flexibility available for routing. A feasible routing solution ensures that track assignments respect these constraints, preventing any region from exceeding its capacity. The following subsections formally define these parameters and their interdependencies.

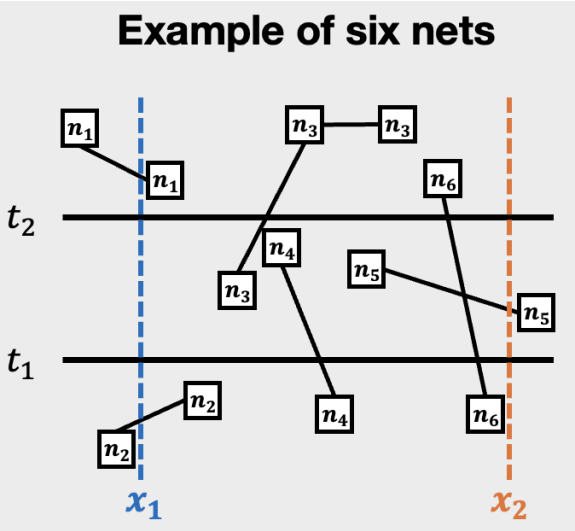


Figure 3.1: Example of six nets

3.1.1 Density

The *density* at a given position x represents the number of nets that overlap with position x. A net is considered to overlap with x if its routing interval spans the coordinate x. This local density reflects the routing demand at each position within the routing region.

Let N ($\subseteq N_{\text{in}}$) be the set of unassigned nets, where each net corresponds to a set of intervals representing the ranges on the tracks it occupies. Similarly, let T ($\subseteq T_{\text{in}}$) be the set of tracks that have not yet been assigned a net. The *density* at position x with respect to the set N is defined as the number of nets in N whose intervals include x. Mathematically, this can be written as:

$$d(x, N) = |\{n \in N \mid x \in I(n)\}| \tag{3.1}$$

where $I(n)$ represents the interval of net n along the track.

The *maximum density* of N is the highest density observed across all positions x, and is given by:

$$D(N) = \max_{x} d(x, N) \tag{3.2}$$

For example, in Fig. 3.1, the density at x_1 and x_2 are $d(x_1) = 2$ and $d(x_2) = 1$, respectively, and the maximum density $D(N) = 2$.

This maximum density represents the peak concentration of nets in the routing region. A higher maximum density indicates a higher level of congestion in the routing channel, which may result in higher routing costs or the need for more tracks to accommodate the nets without overlap.

3.1.2 Capacity

The *capacity* of a set of tracks T is the total number of available tracks in the channel where the nets are to be assigned. A routing track corresponds to a physical path through which a net can be assigned. The capacity is crucial because it limits how many nets can be assigned to tracks without violating the physical constraints of the routing environment.

Mathematically, the capacity of the track set T is represented by the cardinality of T, which is simply the number of tracks in the set:

$$C(T) = |T| \tag{3.3}$$

For example, in Fig. 3.1, there are two tracks are t_1 and t_2 with a capacity of $C = 2$.

This capacity defines the maximum number of nets that can be routed through the available tracks in T. The routing capacity $C(T)$ must be sufficient to accommodate the nets without exceeding the available resources. If the routing demand exceeds the capacity, adjustments must be made either by redistributing the nets or increasing the track set T.

3.1.3 Slack

Slack is a measure of the difference between density and capacity. At each position x, the slack represents the difference between the available capacity at x and the actual density of nets at x. A positive slack indicates that there is sufficient track capacity to accommodate additional nets, while a negative slack means that the density exceeds the available capacity at that position.

The *slack* at position x, given the sets N and T, is defined as:

$$s(x, T, N) = C(T) - d(x, N) \tag{3.4}$$

This equation captures the excess track capacity at position x, considering the current density of nets in N. If the slack is positive, there is available capacity at position x to accommodate more nets. Conversely, if the slack is negative, the routing demand exceeds the available capacity, signaling a potential congestion problem.

For example, in Fig. 3.1, the slack at x_1 and x_2 are $s(x_1) = 0$ and $s(x_2) = 1$, respectively.

The concept of slack is important in routing algorithms because it helps identify regions where track resources are underutilized or overloaded. A routing solution with high slack at all positions ensures that nets can be assigned without violating capacity constraints. However, when slack is negative, adjustments are necessary to redistribute nets or increase track capacity to alleviate congestion.

A feasible track assignment for N within T exists if and only if the maximum density does not exceed the track capacity. This condition can be formally written as:

$$D(N) \leq C(T) \tag{3.5}$$

This condition is referred to as the *density constraint* (DC). If a set of nets N satisfies this constraint for a track set T, the routing can proceed without violating the track capacity at any position.

3.2 Critical Zone

Efficient track assignment is crucial in global routing to minimize vertical wire length and optimize local congestion. A common approach involves greedily assigning tracks to nets sequentially. However, this method requires careful attention to resource constraints, especially the *critical zone* (CZ) within generalized channel. CZ refers to regions where routing demand exceeds available capacity and must be carefully managed during track assignment. It is closely related to the concept of slack, which measures the difference between a track's capacity and local net density. Effectively managing slack within the CZ is key to ensuring a feasible and efficient routing solution.

3.2.1 Definition of Critical Zone

The *Critical Zone* (CZ) for a set of nets N and tracks T is defined as the set of x-coordinates within the routing channel where the available slack is zero or negative, indicating that the density at that position exceeds or matches the track capacity. Mathematically, CZ is expressed as:

$$Z(T, N) = \{x \mid s(x, T, N) \leq 0\}$$

where $s(x, T, N)$ represents the slack at position x in the channel, calculated as the difference between the track's capacity and the number of nets occupying that position. A slack value of zero or lower indicates a lack of available space for additional nets, highlighting a congested area within the routing region. For example, in Fig. 3.2, the slack at x_1, x_2, and x_3 is $s(x_1) = s(x_2) = s(x_3) = 0$, meaning that x_1, x_2, and x_3 belong to CZ.

For effective track assignment, a set of nets $N' \subseteq N$ is said to be CZ-cover if their corresponding trunks fully encompass the entire CZ. In other words, the set N' must include trunks that span all positions in $Z(T, N)$, ensuring that the critical region is sufficiently covered by the nets in N':

$$Z(T, N) \subseteq I(N')$$

where $I(N')$ denotes the union of the trunks corresponding to the nets in N'. These trunks represent the ranges of positions on the tracks occupied by the nets in N'.

For any feasible track assignment $\mathcal{A}$ of nets N to tracks T, it is required that for every position $x \in Z(T, N)$, and for each track $t \in T$, there exists at least one net $n \in N$ such that net n is assigned to track t, and the interval of n includes x. This condition ensures that no position in CZ is left uncovered, thereby guaranteeing that all portions of the routing region are appropriately assigned to tracks:

$$\mathcal{A}(n) = t \quad \text{and} \quad x \in I(n)$$

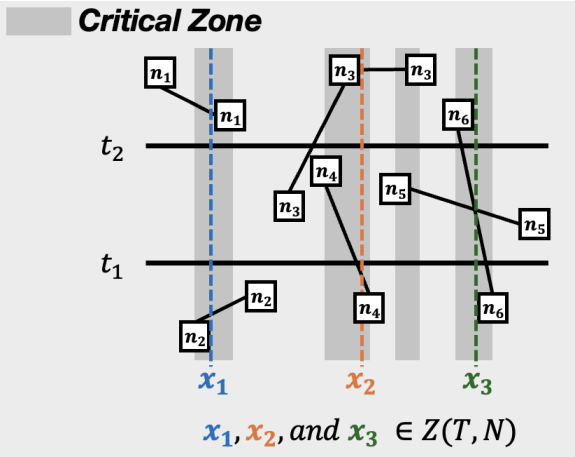


Figure 3.2: Example of Critical Zone

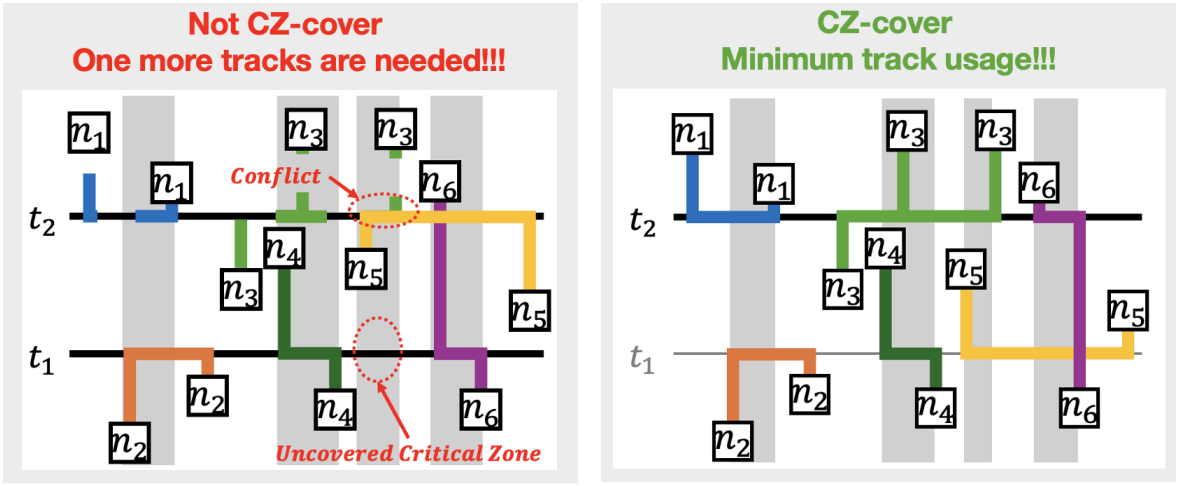
where $\mathcal{A}(n) = t$ means that net n is assigned to track t, and $x \in I(n)$ ensures that the position x lies within the interval of net n.

3.2.2 CZ-cover

A subset $N' \subseteq N$ is considered a *CZ-cover* for N and T if it satisfies two key conditions:

- The trunks of nets in N' cover the entire CZ, such as $Z(T, N) \subseteq I(N')$. This means that the trunks of the nets in N' must span all the positions where slack is non-positive, effectively covering the critical areas of the channel.
- The subset N' satisfies the *horizontal constraint* (HC), which ensures that the nets in N' are positioned in a way that does not violate any horizontal wiring limitations or track capacity restrictions. This is necessary to maintain the feasibility of the track assignments.

Together, these two conditions ensure that the critical areas of the routing path are appropriately addressed and that the track assignments for the remaining nets in $N \setminus N'$ continue to satisfy the overall density constraint (DC) for the remaining tracks. Specifically, if the density constraint is satisfied for the entire set of nets N and tracks T, and N' covers CZs, then assigning the nets in N' to a track $t \in T$ guarantees that the remaining nets in $N \setminus N'$ will still satisfy the density constraint for the reduced set of tracks $T \setminus \{t\}$.



(a) Not CZ-cover (b) CZ-cover

Figure 3.3: Comparison of CZ-cover

Fig. 3.3 illustrates the necessity of the CZ-cover. As shown in Fig. 3.3a, when the CZ is not covered by a net at t_1, an additional track is required to complete the routing. However, if all of CZs are covered by nets, as shown in Fig. 3.3b, no additional track is needed.

In practice, the designation of a *CZ-cover* is crucial for ensuring that critical resources, such as track space, are optimally allocated in the routing process. By addressing the CZ first, the routing algorithm can avoid congestion in the most demanding areas and ensure that the track assignments are efficient and feasible across the entire design space. This approach helps achieve a balanced solution that minimizes the risk of routing violations and ensures high quality, congestion free routing.

3.3 Greedy Routing Framework with Guarantee

The primary goal of GCRP is to complete routing while adhering to strict horizontal capacity constraints. The Greedy Routing Framework with Guarantee (GRFG) [17] is designed to ensure routing completion in GCRP when N_{in} meets the density constraint (DC) for T_{in}, and no vertical constraints exist among the nets in N_{in}. The SDG algorithm, which follows the GRFG, is guaranteed to complete the routing.

To successfully route under the condition where N_{in} satisfies DC for T_{in} and

no vertical constraints exist among the nets, the set of nets assigned to each track in T_{in} must form a CZ-cover. The GRFG introduces the concept of CZ-aware net selection, which ensures that a CZ-cover is established for each track. In this framework, nets are assigned to trunks within each track, starting from a designated position and progressing in both left and right directions, prioritizing coverage of the CZ with higher-priority nets. The pseudo code for the GRFG is provided in Algorithm 1, where, for simplicity, the starting point of each track is set to the leftmost position, meaning that trunks are assigned from left to right within each track. The following discussion is based on this approach.

A set of nets N' is said to be *CZ-aware* in the track assignment of N to T if the following condition holds:

$$[-\infty, x_{\max}(N')] \cap Z(T, N) \subseteq I(N')$$

This means that for any CZ-aware N', the CZ to the left of the interval of a net in N' is covered by $I(N')$. Furthermore, for any point x in the CZ that is not covered by $I(N')$, there exists a net $n \in N \setminus N'$ such that $N' \cup \{n\}$ satisfies the horizontal constraint (HC) and $x \in I(n)$. If a CZ-aware set N' covers the entire CZ, it is also considered a CZ-cover.

In the *CZ-aware net selection* process, for any CZ-aware set N', a net n (where $n \notin N'$) is selected to be added to N' only if the interval $(x_{\max}(N'), x_{\min}(n))$ does not overlap with the CZ, i.e.,

$$(x_{\max}(N'), x_{\min}(n)) \cap Z(T, N) = \emptyset$$

This ensures that the slack is positive for all x-coordinates within the open interval $(x_{\max}(N'), x_{\min}(n))$. CZ-aware net selection guarantees that there is no uncovered CZ to the left of any assigned net, and that $N' \cup \{n\}$ remains CZ-aware.

In step 5 of Algorithm 1, the function `Top_Priority` selects the top-priority net for track t_i among the unassigned nets. The net is not assigned to track t_i if there is an uncovered CZ to the left. This condition is checked by step 6. Note that, even if a net is skipped for assignment to t_i at this stage, it may be assigned later, once lower-priority nets are assigned to track t_i and cover the CZ.

Algorithm 1 GRFG (left-to-right in each track)

Require: set of nets N_{in}, set of tracks $T_{\text{in}} = \{t_1, t_2, \ldots, t_k\}$
Ensure: track assignment $\mathcal{A}(n)$ for all $n \in N_{\text{in}}$

$i \leftarrow 0$, $N \leftarrow N_{\text{in}}$, $T \leftarrow T_{\text{in}}$
while $N \neq \emptyset$ **do**
    $T \leftarrow T \setminus \{t_i\}$, $i \leftarrow i + 1$, $x \leftarrow -\infty$, $U \leftarrow N$
    while $U \neq \emptyset$ **do**
        $n \leftarrow \text{Top\_Priority}(U, t_i)$, $U \leftarrow U \setminus \{n\}$
        if $x \leq x_{\min}(n)$ **and** $(x, x_{\min}(n)) \cap Z(T, N) = \emptyset$ **then**
            $\mathcal{A}(n) \leftarrow t_i$, $x \leftarrow x_{\max}(n)$, $N \leftarrow N \setminus \{n\}$, $U \leftarrow N$
        end if
    end while
end while

An important variation of GRFG is the *Left-Edge* (LE) algorithm, where the priority, referred to as `LEP`, is defined based on the leftmost principle, and nets are assigned track-by-track. It is worth noting that the net selection process in LEP, when choosing the highest priority net from the unassigned nets, always follows CZ-aware net selection.

Fig. 3.4 illustrates an example of the GRFG. The figure shows the set of nets and tracks as input, along with CZ indicated by the shaded region. In this example, priority `SDP` is used for track t_1, as defined in Section 4.3 of Chapter 4.

On the left side of Fig. 3.4, the track assignment without CZ-aware net selection is depicted. First, according to `SDP`, nets n_2, n_4, and n_6 are assigned to track t_1. Then, based on `SDP`, net n_5 is assigned to track t_2. However, due to the GRFG approach, where nets are assigned to trunks from left to right within each track, nets n_1 and n_3 cannot be assigned to track t_2 because net n_5 has already been assigned to t_2, and nets n_1 and n_3 lie to the left of n_5, requiring an additional track.

In track t_1, the uncovered CZ between nets n_4 and n_6 remains, and to prevent a failure, it is essential to ensure that no CZ is left uncovered.

On the right side of Fig. 3.4, the track assignment using the GRFG approach is shown. With CZ-aware net selection, net n_5 is assigned to track t_1 instead of net n_6. In order to complete routing, track t_2 is not initially selected for nets n_6 and n_3, and

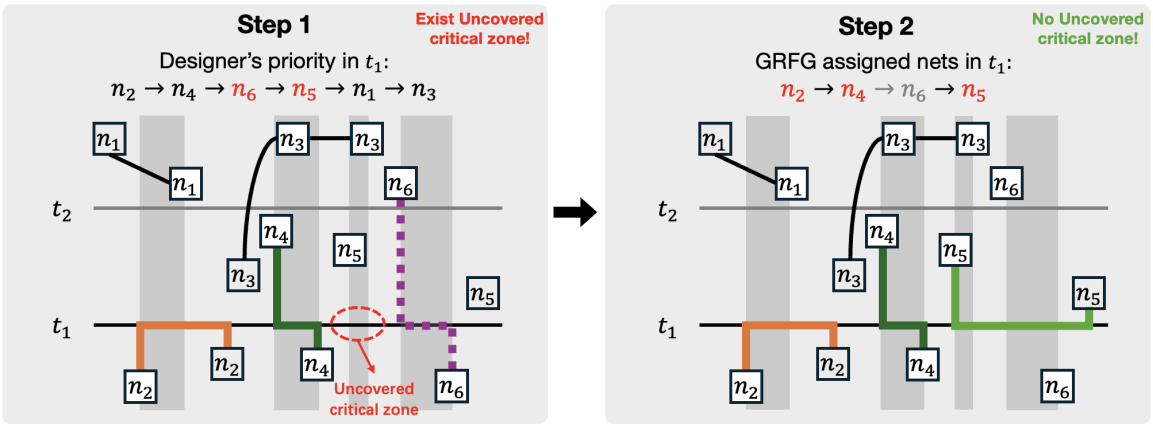


Figure 3.4: Examples of GRFG

net n_1 is assigned first. Subsequently, nets n_3 and n_6 are assigned to track t_2, ensuring that the CZ is covered efficiently.

3.4 Theorem

This section presents a foundational theorem that guarantees the effectiveness of the GRFG in finding a valid track assignment satisfying the HC when the set of input nets N_{in} satisfies the DC for the available tracks T_{in}, and no vertical constraints are imposed.

Theorem. *The number of tracks used by Algorithm 1 to assign nets from N_{in} to T_{in} is at most $|T_{\text{in}}|$, provided that the density of the input nets $D(N_{\text{in}}) \leq |T_{\text{in}}|$ and no vertical constraints exist among the nets.*

Proof. We prove this theorem by contradiction. Assume that there exists a net $n^* \in N_{\text{in}}$ that is not assigned to any track in T_{in} by Algorithm 1. Let $k = |T_{\text{in}}|$ be the number of available tracks. Denote by N_i the set of nets assigned to track t_i for $1 \leq i \leq k$.

We also assume that n^* has the smallest minimum x-coordinate $x_{\min}(n^*) = x_1$ among all nets that have not yet been assigned to a track. This is done without loss of generality.

Since all net trunks are positioned within the k tracks to the left of x_1, and by our assumption that n^* has not yet been assigned, there must exist at least one track

$t_j \in T_{\text{in}}$ that has no net trunk assigned at x_1. Otherwise, if all tracks have net trunks at x_1, it would violate the assumption that $D(N_{\text{in}}) \leq |T_{\text{in}}|$. We choose t_j to be the track with the largest index that satisfies this condition.

Next, consider the set $N = N_j \cup N_{j+1} \cup \cdots \cup N_k$ of nets assigned to tracks t_j through t_k, and the corresponding set of tracks $T = \{t_j, t_{j+1}, \ldots, t_k\}$.

Since the x_1-coordinate is covered by nets already assigned by Algorithm 1, adding n^* would increase the density at x_1, yielding $d(x_1, N) \geq |T|$. This means that the slack at x_1, defined as $s(x_1, T, N) = |T| - d(x_1, N)$, is less than or equal to zero: $s(x_1, T, N) \leq 0$. Therefore, x_1 lies within the CZ for the set of nets N and the tracks T, although it may not yet be within the CZ for N_{in} relative to T_{in}.

Furthermore, since N_j is CZ-aware due to the net selection process in Algorithm 1, no net trunks from N_j extend beyond x_1 on track t_j, because x_1 is critical. Let $x_0 = x_{\max}(N_j)$ represent the maximum x-coordinate of the nets in N_j. By the definition of N_j, we have $x_0 < x_1$.

Now, consider the interval (x_0, x_1). For any x in this interval, the slack $s(x, T, N) > 0$, meaning there is positive slack available to assign nets in this region. Therefore, net n^* satisfies the conditions required for assignment to track t_j. Specifically, the interval (x_0, x_1) does not intersect the CZ for the set N and the track set T, i.e., $(x_0, x_1) \cap Z(T, N) = \emptyset$, ensuring that n^* can be assigned to track t_j without violating any constraints.

This contradicts the assumption that n^* is unassigned, as it would be assigned to track t_j based on the selection criteria of Algorithm 1. Since there are no vertical constraints, we conclude that n^* must be assigned, proving that the number of tracks used by Algorithm 1 is at most $|T_{\text{in}}|$.

Thus, the theorem is proven by contradiction. □

This theorem guarantees that GRFG will always find a valid track assignment satisfying HC, as long as the input nets meet DC and there are no vertical constraints. This provides a solid foundation for the correctness and efficiency of the GRFG algorithm in routing problems.

3.5 Summary

This chapter provides an in depth exploration of the development of an efficient routing framework specifically designed for GCRP, with a focus on managing constrained horizontal capacity. The primary aim is to optimize the routing process by minimizing track usage while ensuring the maximum number of nets are successfully routed within the available resources.

To accomplish this objective, the chapter introduces several key concepts: density, capacity, slack, and critical zones, which are crucial for assessing routing feasibility and guiding the routing procedure. These measures serve as the foundation for developing strategies to handle routing constraints effectively.

The chapter begins by defining density, capacity, and slack within the context of GCRP, establishing their roles in determining routing efficiency and feasibility. *Density* refers to the number of nets that pass through a specific coordinate, reflecting the congestion at that point. *Capacity* is defined as the total number of tracks available within the routing channel, serving as a limiting factor for net assignments. *Slack*, on the other hand, represents the difference between the available track capacity and the actual density at a given point, offering crucial margins for adjusting track assignments and mitigating congestion.

A pivotal concept introduced is the *Critical Zone* (CZ), which identifies regions in the routing channel where the available tracks are fully utilized, leading to a bottleneck in the routing process. The CZ is a key concept for efficient track assignments, particularly when the routing resources are under heavy constraint. By understanding and managing these zones, the routing framework can optimize track usage and avoid overloading any part of the channel.

The Greedy Routing Framework with Guarantee (GRFG) is then presented in detail, offering a systematic approach to routing that guarantees effectiveness under conditions where the density constraints are met and no vertical constraints are imposed. GRFG employs an innovative strategy known as *CZ-aware net selection*, where nets are assigned based on their interaction with CZs, ensuring that track resources are used efficiently. This approach allows the framework to prioritize the most critical nets, ensuring they are routed effectively while avoiding congestion in the most heavily utilized regions.

A formal theorem underpinning GRFG is also provided, which mathematically proves that the number of tracks used by the algorithm will not exceed the available tracks, as long as the density constraints are satisfied and no vertical constraints are imposed. The proof of this theorem establishes the efficiency and completeness of the GRFG, ensuring that the routing process will always be feasible within the given constraints.

In summary, this chapter lays the groundwork for an efficient routing strategy for GCRP by introducing essential concepts such as density, capacity, slack, and CZs, and presenting a robust routing framework in the form of GRFG. The theoretical guarantee provided by the theorem ensures that the framework is not only effective but also operates within the available resources, offering a reliable solution for routing under constrained conditions.

Chapter 4

Length Minimization Algorithm of GCRP

Minimizing total wire length is a key objective in VLSI routing, as it directly impacts circuit performance, power consumption, and manufacturability. Excessive wire length increases parasitic resistance and capacitance, leading to higher signal delays, degraded timing, and reduced operational speed. Longer wires also contribute to greater power dissipation, exacerbating thermal and reliability concerns in densely packed chips.

From a fabrication perspective, reducing wire length improves routability by alleviating congestion and minimizing design rule violations, thus enhancing the feasibility of a successful routing solution. Given these critical considerations, this chapter presents an efficient algorithm for GCRP with length minimization, optimizing wire length while adhering to essential routing constraints. The proposed approach systematically reduces unnecessary wire length, ensuring efficient utilization of routing resources and achieving high-performance routing solutions.

GCRP is formulated as an assignment problem, where the input consists of a set of nets $N_{\text{in}} = \{n_1, n_2, \ldots, n_m\}$ and a set of horizontal tracks $T_{\text{in}} = \{t_1, t_2, \ldots, t_k\}$ within a given channel. A net is defined by a set of pins that must be interconnected via wire segments. The spatial location of a pin p is denoted by $(x(p), y(p))$. For any net n, the minimum and maximum x-coordinates of its pins are given by:

$$x_{\min}(n) = \min_{p \in n} x(p), \quad x_{\max}(n) = \max_{p \in n} x(p).$$

Similarly, the minimum and maximum y-coordinates are denoted as $y_{\min}(n)$ and $y_{\max}(n)$, respectively. The horizontal span and total vertical displacement of a net n are defined as:

$$d^x(n) = x_{\max}(n) - x_{\min}(n), \quad d^y(n) = \sum_{p \in n} |y(p) - y(p_{\text{mid}})|,$$

where p_{mid} represents the pin with the median y-coordinate, specifically the $\lceil |n|/2 \rceil$-th largest among the pins of net n. In the case of STST, $d^x(n)$ corresponds to the horizontal span, while $d^y(n)$ provides a lower bound on the cumulative vertical stretch of the net.

The interval associated with a net n is defined as:

$$I(n) = [x_{\min}(n), x_{\max}(n)].$$

For a set of nets N, the union of their x-coordinate intervals is:

$$I(N) = \{x \mid x \in I(n),\, n \in N\},$$

and the maximum x-coordinate in N is:

$$x_{\max}(N) = \max_{p \in n \in N} x(p).$$

By convention, if N is empty, then $x_{\max}(N) = -\infty$.

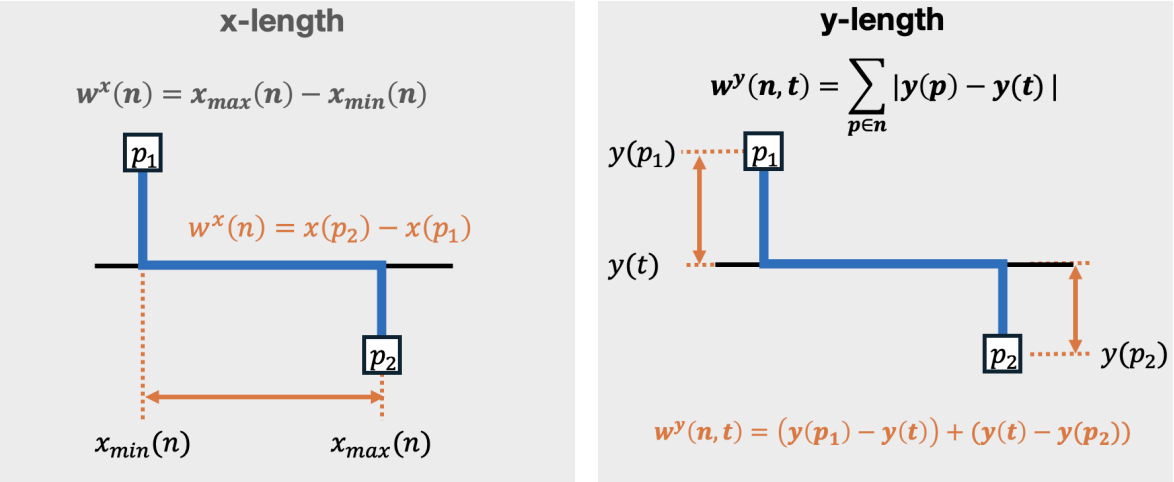
A horizontal constraint exists between two nets n_i and n_j if and only if their intervals intersect, i.e.,

$$I(n_i) \cap I(n_j) \neq \emptyset.$$

In the context of STST, it is required that the trunks of connected nets be assigned to tracks. Typically, a vertical constraint arises when two pins, $p_1 \in n_1$ and $p_2 \in n_2$, have the same x-coordinate, such as $x(p_1) = x(p_2)$. However, this work assumes no vertical constraints, even in cases where the x-coordinates of pins overlap.

4.1 Definition of Length in GCRP

In GCRP, the length of a net determines its contribution to overall routing complexity and congestion. When a net n is assigned to a track t, its total wire length is defined



(a) X-Length

(b) Y-Length.

Figure 4.1: Definition of Length

as

$$w(n,t) = w^x(n) + w^y(n,t), \tag{4.1}$$

where $w^x(n)$ represents the horizontal length and $w^y(n,t)$ represents the vertical length. These components are defined as follows:

$$w^x(n) = x_{\max}(n) - x_{\min}(n), \tag{4.2}$$

$$w^y(n,t) = \sum_{p \in n} |y(p) - y(t)|, \tag{4.3}$$

where $x_{\max}(n)$ and $x_{\min}(n)$ denote the maximum and minimum x-coordinates of net n, respectively, and $y(t)$ is the y-coordinate of track t. Figs. 4.1a and 4.1b illustrate the definitions of x-length and y-length, respectively.

Since the horizontal length $w^x(n)$ is determined solely by the net's span in the horizontal direction and remains unchanged during track assignment, the focus of this chapter is on minimizing the total vertical wire length $w^y(n,t)$, which directly impacts congestion and wire resource utilization.

A track assignment $\mathcal{A}$ is a function that assigns a net n to a specific track t, denoted as $\mathcal{A}(n) = t$. The total vertical wire length for a set of nets N under a given track assignment $\mathcal{A}$ is then given by:

$$w^y(N, \mathcal{A}) = \sum_{n \in N} w^y(n, \mathcal{A}(n)). \tag{4.4}$$

This metric is crucial in evaluating vertical congestion and optimizing resource distribution across available tracks.

The objective of GCRP in this chapter is to minimize the total vertical wire length while ensuring that all nets in the given set N_{in} are assigned to tracks in T_{in} while satisfying horizontal constraints. This is formally expressed as the following optimization problem:

$$\begin{aligned} \min \quad & w^y(N_{\text{in}}, \mathcal{A}) = \sum_{n \in N_{\text{in}}} w^y(n, \mathcal{A}(n)), \\ \text{s.t.} \quad & \mathcal{A}(n) \in T_{\text{in}}, \quad \forall n \in N_{\text{in}}, \\ & I(n_i) \cap I(n_j) = \emptyset \text{ if } \mathcal{A}(n_i) = \mathcal{A}(n_j), \quad \forall n_i, n_j \in N_{\text{in}}. \end{aligned} \tag{4.5}$$

This formulation ensures that all nets are assigned to feasible tracks while preventing horizontal overlap violations. The next sections explore algorithmic strategies to achieve this objective efficiently.

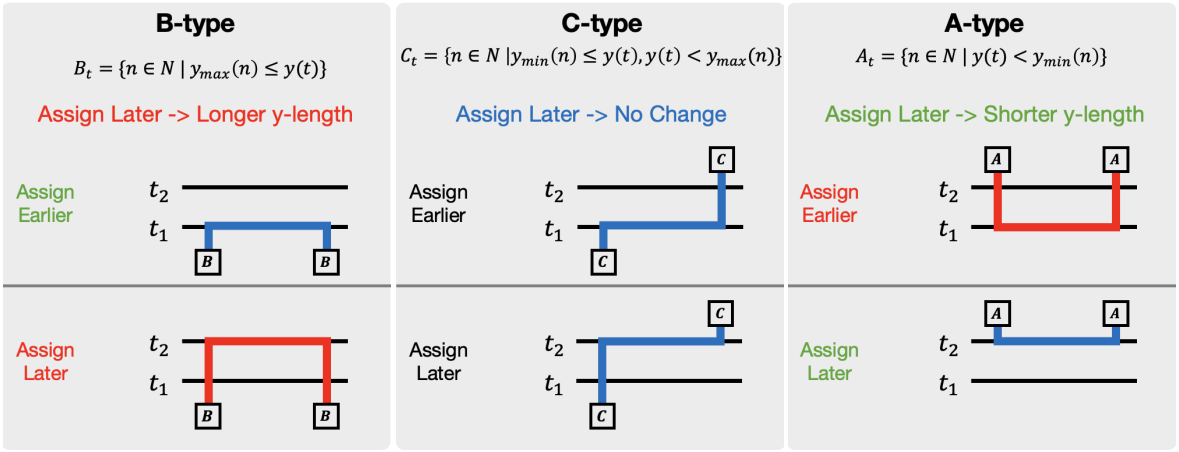
4.2 Algorithm for Two-Pin Nets

This research begins by focusing on two-pin nets, where the total wire length is given by:

$$w^y = |y(p_1) - y(t)| + |y(p_2) - y(t)|.$$

To efficiently minimize the total wire length of two-pin nets, this work introduces the **BCA (Below, Cross, Above)** channel routing algorithm [21]. The algorithm assigns nets based on BCA priority, while minimizing track usage. It iteratively assigns nets from the input set N_{in} to the track set T_{in}, ensuring compliance with the density constraint (DC) when $D(N_{\text{in}}) \leq |T_{\text{in}}|$. High-priority nets that satisfy both horizontal and density constraints are given precedence.

Track assignment progresses from bottom to top, with minimal impact on wire length. The BCA algorithm mitigates this by strategically prioritizing nets. While the initial track assignment may not yield a locally optimal solution due to trunk shifts and swaps, post-processing techniques can be applied to refine the results further.



(a) B-type net. (b) C-type net. (c) A-type net.

Figure 4.2: Classification of nets based on the BCA priority scheme.

4.2.1 BCA Priority

The priority of a net in the BCA algorithm is determined by two factors:

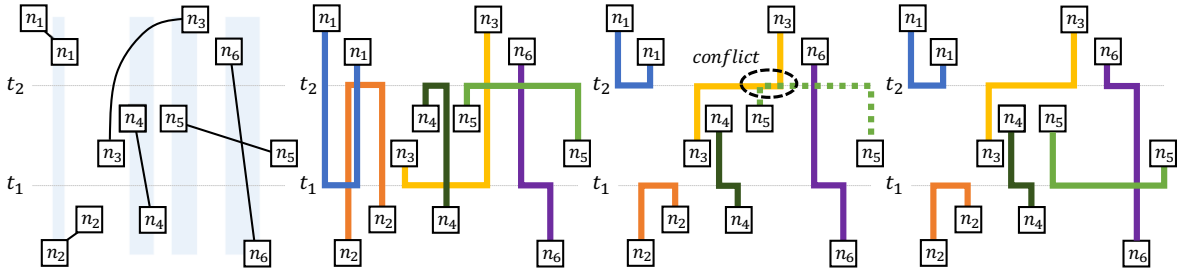
1. The relative position of a net's pins to the track, categorized as *Below* (B), *Cross* (C), or *Above* (A).
2. The ascending order of $y_{\max}(n)$ within each category.

Given a track t, nets are classified as follows:

$$\begin{aligned} B_t &= \{n \in N \mid y_{\max}(n) \leq y(t)\}, \\ C_t &= \{n \in N \mid y_{\min}(n) \leq y(t) < y_{\max}(n)\}, \\ A_t &= \{n \in N \mid y(t) < y_{\min}(n)\}. \end{aligned} \tag{4.6}$$

The BCA algorithm processes nets in the order $B_t \rightarrow C_t \rightarrow A_t$, further prioritizing within each category based on ascending values of $y_{\max}(n)$.

Fig. 4.2 illustrates the BCA classification. In Fig. 4.2a, the net is classified as type B for both tracks t_1 and t_2, demonstrating that assigning B-type nets early minimizes wire length. In Fig. 4.2b, the net is type C for both tracks t_1 and t_2, indicating that C-type nets may remain in their category across multiple tracks. In Fig. 4.2c, the net is type A for t_1 and t_2, showing that assigning A-type nets later results in shorter wire lengths.



(a) Input. (b) Left-Edge $w^y = 38$. (c) BCA priority only. (d) BCA $w^y = 22$.

Figure 4.3: Assignment comparison.

4.2.2 BCA Channel Routing Algorithm

Algorithm 2 outlines the BCA routing procedure, which uses BCA priority for wire length minimization and incorporates CZ-aware net selection to ensure feasible assignments. While the initial assignments may not be locally optimal due to trunk shifts and swaps, post-processing can further refine the results.

For instance, consider the six-net example shown in Fig. 4.3a. The priority order for track t_1 is:

$$n_2^B, n_4^C, n_6^C, n_5^A, n_1^A, n_3^A.$$

For track t_2, the priority order is:

$$n_2^B, n_4^B, n_5^B, n_6^C, n_3^C, n_1^A.$$

Fig. 4.3c demonstrates a case where applying only BCA priority, without considering GRFG, leads to infeasible assignments. Initially, nets n_2, n_4, and n_6 are assigned to track t_1. However, this assignment fails since net n_5 cannot be assigned on track t_2, necessitating an additional track. In this case, a CZ remains between n_4 and n_6 on track t_1. The GRFG mechanism resolves this by ensuring no uncovered CZ exists to the left of a net at the time of assignment.

By integrating CZ-aware net selection and conflict resolution mechanisms, the BCA routing algorithm achieves both optimality and feasibility in net assignments. The proposed method effectively minimizes wire length while ensuring constraint compliance, demonstrating superior performance compared to conventional approaches.

Algorithm 2 BCA routing

Require: set of nets N_{in}, set of tracks $T_{\text{in}} = \{t_1, t_2, \ldots, t_k\}$
Ensure: track assignment $\mathcal{A}(n)$ for all $n \in N_{\text{in}}$

```
1: i ← 0, N ← N_in, T ← T_in
2: while N ≠ ∅ do
3:     T ← T \ {t_i}, i ← i + 1, x ← −∞, U ← N
4:     while U ≠ ∅ do
5:         n ← Top_BCA_Priority(U, t_i), U ← U \ {n}
6:         if x ≤ x_min(n) and (x, x_min(n)) ∩ Z(T, N) = ∅ then
7:             A(n) ← t_i, x ← x_max(n), N ← N \ {n}, U ← N
8:         end if
9:     end while
10: end while
```

4.3 Algorithm for Multi-Pin Nets

For multi-pin nets, the *Symmetric Difference General* (SDG) channel routing algorithm is proposed to minimize the total vertical wire length [22]. SDG follows a greedy heuristic approach, making locally optimal choices at each step in an attempt to achieve a near optimal global solution. While SDG does not guarantee absolute optimality, it effectively reduces vertical wire length by leveraging the *Symmetric Difference (SD)* metric to guide track assignments.

4.3.1 Symmetric Difference (SD) of Pins of a Net

The vertical wire length of a net is determined by the track to which it is assigned. When all pins of a net n are interconnected using STST, the total vertical wire length is defined as:

$$w^y(n,t) = \sum_{p \in n} |y(p) - y(t)| \tag{4.7}$$

where t represents the assigned track and $y(p)$ denotes the vertical coordinate of pin p.

To quantify the balance of a net's pins around a given track, SD of net n at track t is defined as:

$$\texttt{SD}(n,t) = |\{p \in n \mid y(p) < y(t)\}| - |\{p \in n \mid y(p) > y(t)\}|. \tag{4.8}$$

This value represents the difference between the number of pins positioned below and above the assigned track. A smaller absolute value of $\texttt{SD}(n,t)$ corresponds to a more balanced pin distribution, thereby reducing the total vertical wire length. The optimal vertical wire length is achieved when $\texttt{SD}(n,t) = 0$, meaning the number of pins above and below the track is equal. Therefore, minimizing $|\texttt{SD}(n,t)|$ serves as a crucial guiding principle for effective track assignment.

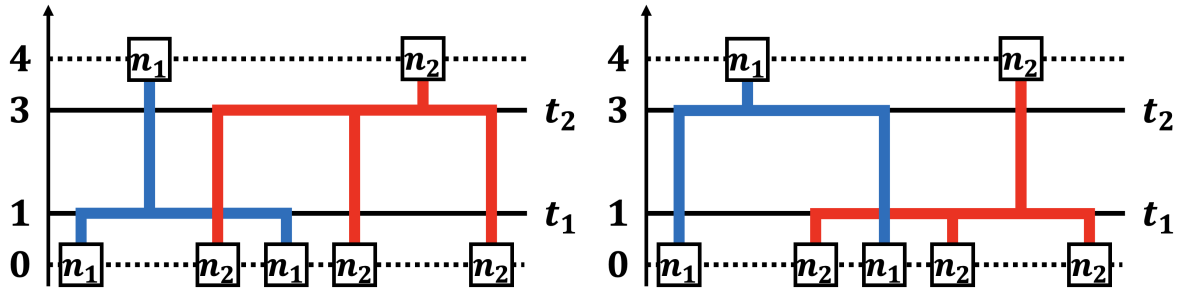
To illustrate the role of SD in track assignment, consider the case of two nets, n_1 and n_2, and their possible assignments to tracks t_1 and t_2, as depicted in Figure 4.4. The corresponding vertical wire lengths for different assignments are as follows: $w^y(n_1,t_1) = 5, w^y(n_1,t_2) = 7, w^y(n_2,t_1) = 6, w^y(n_2,t_2) = 10$.

Although assigning either net to t_1 yields a lower individual vertical wire length, HC prevent both nets from occupying the same track. The global optimization perspective is required to minimize the total vertical wire length. Assigning n_2 to t_1 (Figure 4.4(b)) results in a lower total vertical wire length compared to assigning n_1 to t_1 (Figure 4.4(a)).

The increase in vertical wire length when a net is shifted from a lower track t to an upper track t' is directly proportional to $\texttt{SD}(n,t)$, assuming $\texttt{SD}(n,t) = \texttt{SD}(n,t')$. This observation leads to the following heuristic strategy for minimizing total vertical wire length:

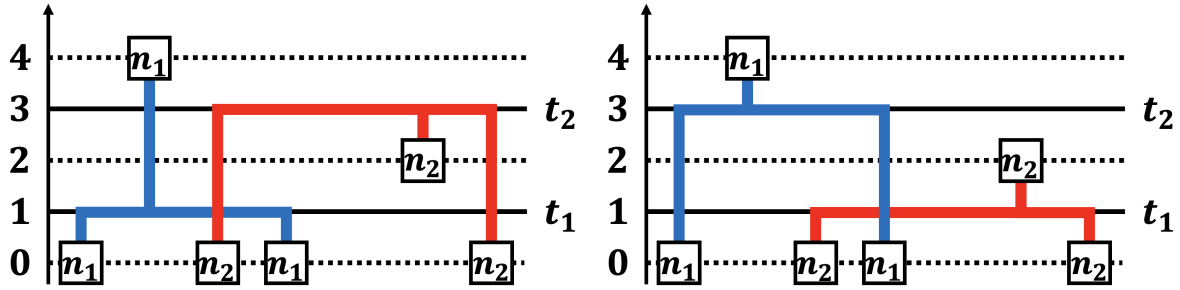
- **Nets with larger SD values should be assigned to lower tracks**, as this placement minimizes their vertical extension.
- **Nets with smaller SD values can be placed at higher tracks**, as they inherently contribute less to the overall vertical wire length.

By prioritizing nets with large $|\texttt{SD}|$ for lower tracks, the SDG algorithm systematically reduces the aggregate vertical wire length, achieving an efficient routing solution.



(a) $w^y = 15$ (b) $w^y = 13$

Figure 4.4: $\texttt{SD}(n_1, t_1) = 1 < \texttt{SD}(n_2, t_1) = 2$, $\texttt{SD}(n_1, t_2) = 1$, $\texttt{SD}(n_2, t_2) = 2$



(a) $w^y = 12$ (b) $w^y = 10$

Figure 4.5: $\texttt{SD}(n_1, t_1) = \texttt{SD}(n_2, t_1) = 1$, $\texttt{SD}(n_1, t_2) = 1 < \texttt{SD}(n_2, t_2) = 3$

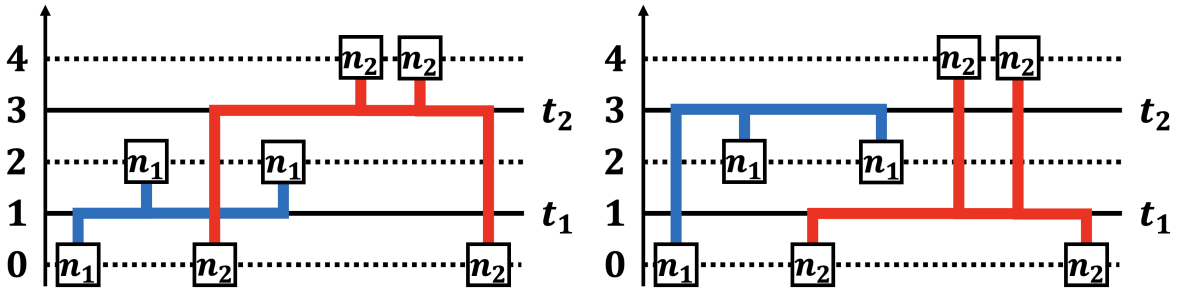
In Fig. 4.5, we compare two track assignments for nets n_1 and n_2 on tracks t_1 and t_2, with vertical wire lengths:

$$w^y(n_1, t_1) = 5, \quad w^y(n_1, t_2) = 7,$$
$$w^y(n_2, t_1) = 3, \quad w^y(n_2, t_2) = 7.$$

Assigning n_2 which has a larger SD for t_2 to t_1 minimizes the total vertical wire length, as shown in Fig. 4.5(b). Conversely, assigning n_1 to t_1, shown in Fig. 4.5(a), results in a higher total vertical wire length, even though both nets have the same SD for t_1. This suggests that nets should be assigned lexicographically to lower tracks based on their SD-sequence $(\texttt{SD}(n, t_1), \texttt{SD}(n, t_2), \ldots, \texttt{SD}(n, t_m))$, given $y(t_1) < y(t_2) < \cdots < y(t_m)$.

Similarly, Fig. 4.6 evaluates another assignment scenario with vertical wire lengths:

$$w^y(n_1, t_1) = 3, \quad w^y(n_1, t_2) = 5,$$
$$w^y(n_2, t_1) = w^y(n_2, t_2) = 8.$$



(a) $w^y = 11$ (b) $w^y = 13$

Figure 4.6: $\texttt{SD}(n_1, t_1) = -1 < \texttt{SD}(n_2, t_1) = 0$, $\texttt{SD}(n_1, t_2) = 3 > \texttt{SD}(n_2, t_2) = 0$

Assigning n_1, which has a smaller SD for t_1, to t_1 yields a lower total vertical wire length, as illustrated in Fig. 4.6(a), compared to placing n_2 on t_1, as shown in Fig. 4.6(b). This example demonstrates that the previously suggested strategy does not always guarantee an optimal solution, highlighting the complexity of net assignment optimization.

4.3.2 Priority by Symmetric Difference

This subsection introduces the prioritization of nets in set N for track t_i $(1 \leq i \leq k)$, given that $y(t_i) < y(t_{i+1}) < \cdots < y(t_k)$.

The SD-sequence for a net n on track t_i is defined as:

$$(\texttt{SD}(n, t_i), \texttt{SD}(n, t_{i+1}), \ldots, \texttt{SD}(n, t_k)).$$

As noted earlier, assigning nets with larger SD value for track t_i in descending lexicographical order is advantageous when $y(t_i) < y(t_{i+1}) < \cdots < y(t_k)$. The priority SDP for track t_i is therefore determined by this descending lexicographical order, with ties resolved arbitrarily using the net index.

For example, in Fig. 4.7, the SD-sequence of nets n_1, n_2, n_3, n_4, n_5, and n_6 on track t_1 are $(-2, -2)$, $(2, 2)$, $(-3, -1)$, $(0, 2)$, $(-2, 2)$, and $(0, 0)$, respectively. For track t_2, the SD-sequence are (-2), (2), (-1), (2), (2), and (0), respectively. Consequently, the SDPs for tracks t_1 and t_2 are $(n_2, n_4, n_6, n_5, n_1, n_3)$ and $(n_2, n_4, n_5, n_6, n_3, n_1)$, respectively.

In Fig. 4.5, the SD-sequence of nets n_1 and n_2 on track t_1 are $(\texttt{SD}(n_1, t_1), \texttt{SD}(n_1, t_2)) =$

Algorithm 3 SDG Channel Routing Algorithm

Require: set of nets N_{in}, set of tracks $T_{\text{in}} = \{t_1, t_2, \ldots, t_m\}$ where $y(t_1) < y(t_2) < \ldots < y(t_m)$

Ensure: track assignment $\mathcal{A}(n)$ for all $n \in N_{\text{in}}$

$i \leftarrow 0$, $N \leftarrow N_{\text{in}}$, $T \leftarrow T_{\text{in}}$
while $N \neq \emptyset$ **do**
    $T \leftarrow T \setminus \{t_i\}$, $i \leftarrow i + 1$, $x \leftarrow -\infty$, $U \leftarrow N$
    while $U \neq \emptyset$ **do**
        $n \leftarrow \text{Largest\_SD}(U, t_i)$, $U \leftarrow U \setminus \{n\}$
        if $(x, \infty) \cap Z(T, N) = \emptyset$ **and** $\texttt{SD}(n, t_i) < 0$ **then**
            break
        end if
        if $x \leq x_{\min}(n)$ **and** $(x, x_{\min}(n)) \cap Z(T, N) = \emptyset$ **then**
            $\mathcal{A}(n) \leftarrow t_i$, $x \leftarrow x_{\max}(n)$, $N \leftarrow N \setminus \{n\}$, $U \leftarrow N$
        end if
    end while
end while

$(1, 1)$ and $(\texttt{SD}(n_2, t_1), \texttt{SD}(n_2, t_2)) = (1, 3)$, respectively. The `SDP` for track t_1 is therefore (n_2, n_1).

In prior work [21], the BCA algorithm was introduced for two-pin nets, classifying them into three categories Below, Cross, and Above based on their track assignments, corresponding to SDs of 2, 0, and -2, respectively. Thus, the `BCA` priority used in the BCA algorithm is a special case of `SDP`.

4.3.3 SDG Channel Routing Algorithm

The Symmetric Difference General (SDG) channel routing algorithm, presented in Algorithm 3, is an efficient method for assigning nets within a channel. This algorithm aims to accomplish routing when the given set of nets meets DC and no vertical constraints are enforced, and to minimize length by GRFG with the priority derived by the symmetric difference of pins of a net.

In Algorithm 3, tracks are denoted as $T_{\text{in}} = \{t_1, t_2, \ldots, t_m\}$, with the condition that the vertical positions of the tracks satisfy $y(t_1) < y(t_2) < \cdots < y(t_m)$. The algorithm

processes the nets and assigns them to tracks iteratively.

Step 5 uses the function `Largest_SD` to identify the net with the highest priority for track t_i from the unassigned nets. This is determined by SD-sequence, which prioritizes nets according to the top net in the symmetric difference priority (SDP) for track t_i.

The core of the algorithm relies on the GRFG approach, which assigns nets to tracks starting from the lowest track, progressing upward one by one. This sequence of assignments ensures that nets are placed on tracks that adhere to HC of the routing process. As a result, nets placed on lower tracks tend to exhibit smaller vertical wire lengths. Specifically, a net n will have a smaller vertical wire length when assigned to track t_i if the absolute value of its symmetric difference $|\texttt{SD}(n, t_i)|$ is minimized.

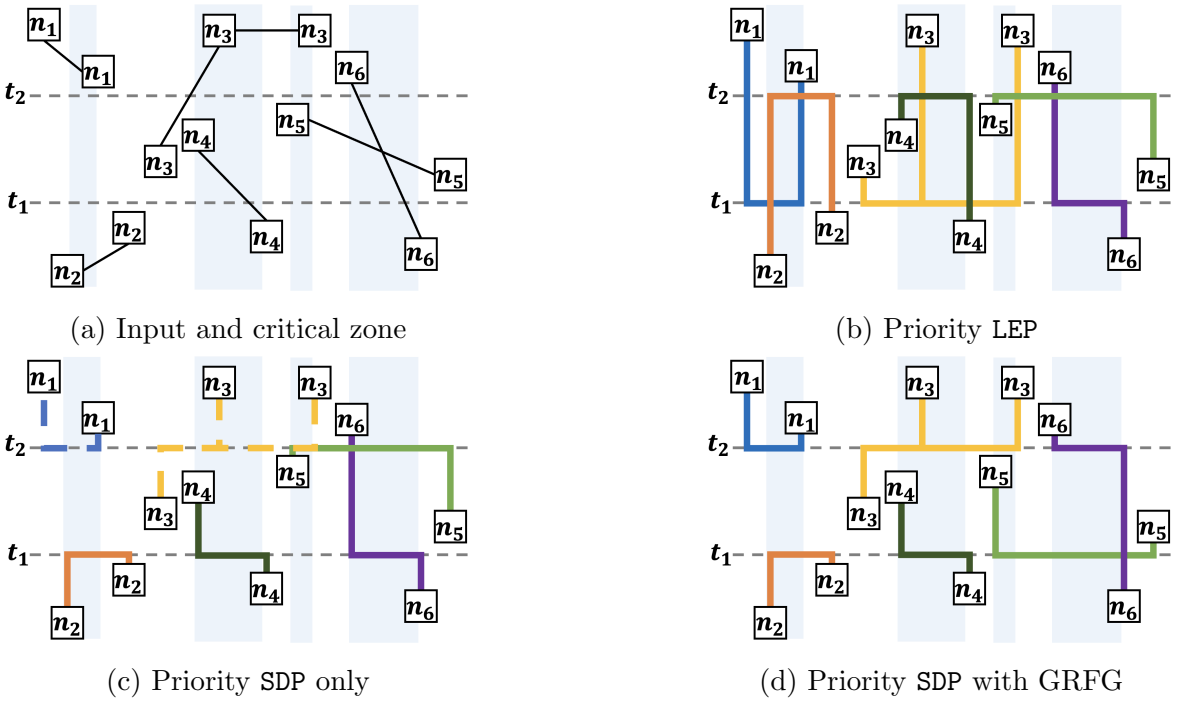
However, the assignment to higher tracks is inherently associated with larger SD, which can increase the vertical wire length. Despite this, if SD value of a net for a track is negative ($\texttt{SD}(n, t_i) < 0$), assigning it to a higher track might actually reduce the overall vertical wire length. Therefore, Algorithm 3 incorporates a flexible approach that considers net priority in combination with the potential reduction in vertical wire length for specific assignments.

Track assignment is suspended when step 6 is reached, indicating that a CZ for the track has been adequately covered, and no further assignment of nets with non-negative SD values can occur without violating horizontal constraints. This ensures that the track assignment process respects both vertical and horizontal constraints while striving to minimize the total vertica wire length.

Additionally, the algorithm ensures that the final track assignments are optimal by continuously updating the set of unassigned nets and revisiting previous decisions as necessary. The SDG algorithm is capable of efficiently handling complex routing tasks while maintaining the integrity of the routing design by strictly adhering to the horizontal constraints and minimizing the vertical wire length.

4.4 Algorithm for Post Processing

While the SDG algorithm offers an efficient method for channel routing, it remains a heuristic approach that does not guarantee the optimal total vertical wire length.



(a) Input and critical zone

(b) Priority `LEP`

(c) Priority `SDP` only

(d) Priority `SDP` with GRFG

Figure 4.7: Track assignments by SDG.

To address this limitation, this research introduce a post-processing algorithm (Algorithm 4) designed to iteratively reduce the total vertical wire length in a greedy manner. This algorithm refines the initial routing solution produced by SDG, aiming to achieve a local minimum for vertical wire length.

The post-processing algorithm begins by initializing the total vertical wire length, denoted as w, and setting a flag to indicate whether further improvements are possible. The algorithm iterates over all nets, $n \in N_{\text{in}}$, and attempts to reduce the total vertical wire length by shifting or exchanging nets between tracks.

For each net n with a non-zero current SD value, the algorithm examines all tracks t where the absolute SD value of n is smaller than its current SD value on the assigned track, $a(n)$. If there are no overlaps between the interval $I(n)$ and the intervals of nets already assigned to track t, the algorithm attempts to shift net n to track t. This `shift` operation is performed only if the new track assignment, denoted as a', results in a reduction of the total vertical wire length, $w(a') < w$, as shown in Fig. 4.8(a). If the shift is successful, the total vertical wire length is updated, and the flag is set to true to signal that further improvements are possible.

Algorithm 4 Post-Processing for SDG

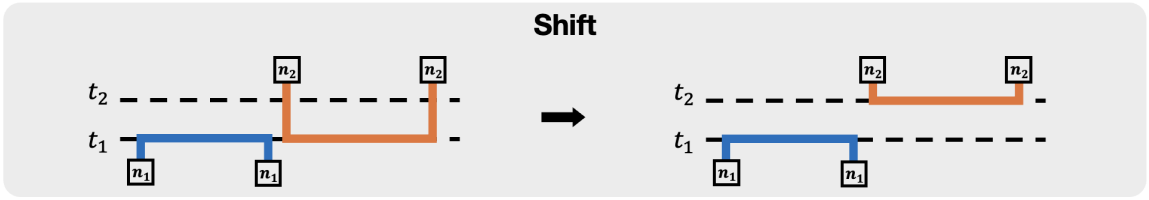
Require: track assignment $a(n)$ for all $n \in N_{\text{in}}$
Ensure: track assignment $a(n)$ for all $n \in N_{\text{in}}$

$w \leftarrow w(a)$, *flag* $\leftarrow$ T
while *flag* == T **do**
    flag $\leftarrow$ F
    for all $n \in N$ **do**
        for all $t \in \{t \in T_{\text{in}} \mid |SD(n,t)| < |SD(n,a(n))|\}$ **do**
            if $I(n) \cap I(\{n' \mid a(n') = t\}) = \emptyset$ **then** ▷ try shift
                $a' \leftarrow \text{shift}(a, n, t)$
                if $w(a') < w$ **then**
                    $w \leftarrow w(a')$, $a \leftarrow a'$, *flag* $\leftarrow$ T ▷ update
                end if
            else ▷ try exchange
                $a' \leftarrow \text{exchange}(a, n, t)$
                if feasible(a') **and** $w(a') < w$ **then**
                    $w \leftarrow w(a')$, $a \leftarrow a'$, *flag* $\leftarrow$ T ▷ update
                end if
            end if
        end for
    end for
end while

In cases where shifting is not feasible due to overlapping intervals, the algorithm considers exchanging net n with another net n' currently assigned to track t. This `exchange` operation is performed only if the new assignment, a', satisfies all feasibility constraints and results in a reduction of the total vertical wire length. Examples in Fig. 4.8(b) and (c) show the improved solution by exchange two nets and exchange two tracks respectively.

The algorithm continues to iterate until no changes occur during a complete iteration, at which point it terminates. The resulting track assignments are locally optimal with respect to the vertical routing length.

The effectiveness of this post-processing algorithm is demonstrated in the example shown in Fig. 4.6, where the improved solution is presented in Fig. 4.6(a).This solution



(a) Shift

(b) Exchange Two nets

(c) Exchange Two tracks

Figure 4.8: Example of Shift and Exchange.

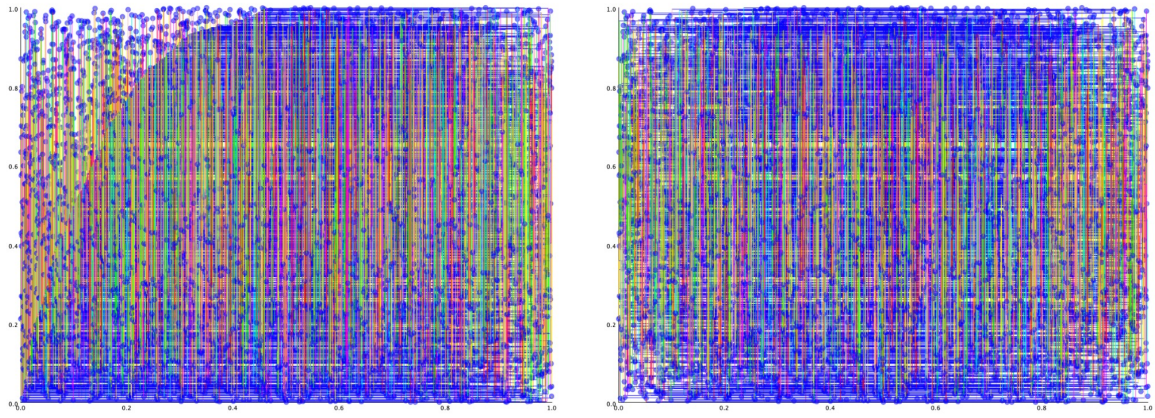
provides a better routing outcome compared to the original SDG solution shown in Fig. 4.6(b).

4.5 Experimental Results

4.5.1 Experimental Setup

We conducted experiments to evaluate the performance of the SDG algorithm compared to other methods. These experiments were implemented in Java 21.0.2 and executed on an Apple M1 Pro CPU, with the largest benchmark tasks completed in under 5 minutes.

All benchmarks were generated randomly. Pin locations for each net were uniformly distributed, with $x(p), y(p) \sim \mathcal{U}(0, 1)$ for all pins $p \in n$. Similarly, y-coordinates



(a) Left-Edge ($w^y = 1937.1$) (b) SDG ($w^y = 1377.4$)

Figure 4.9: Results on gm-5 (#net=1000, $D(N_{\text{in}}) = 4439$).

for the $D(N_{\text{in}})$ tracks were randomly generated using the same uniform distribution. The number of pins per net in the multi-pin benchmarks ranged from 2 to 10, chosen randomly.

We generated two sets of random benchmarks for evaluating GCRP: two-pin benchmarks, prefixed with "gt-", and multi-pin benchmarks, prefixed with "gm-". The results for the two-pin benchmarks are presented in Tables 4.1 and 4.2, while the results for the multi-pin benchmarks are shown in Tables 4.3 and 4.4.

In the tables, the entries "LE", "BCA", "SDG w/o step6", "SDG", and "PP" correspond to the Left-Edge algorithm [5], BCA [21], SDG excluding step 6, the complete SDG, and SDG with post-processing, respectively. For multi-pin nets, BCA is enhanced by categorizing nets as Below, Cross, and Above based on their SD values (positive, zero, and negative, respectively).

"Distance" refers to the total x-distance $\sum_n d^x(n)$ and total y-distance of nets $\sum_n d^y(n)$. "y-length" denotes the total y-length of nets, with the total y-distance used as a reference. "Time" represents the computation time, with SDG post-processing serving as the benchmark.

4.5.2 Comparison with Conventional Methods

The experimental results show that SDG outperforms other methods in terms of total y-length in most scenarios. SDG produces effective solutions in a relatively short time, which can be further refined with post-processing. These results validate the efficacy of the `SDP` priority in SDG and the benefits of the deferral approach used in step 6.

As the input size increases, the proportion of computation time devoted to post-processing grows. Multi-pin benchmarks, with densities around 90%, present greater constraints compared to two-pin benchmarks. This is due to the difficulty of assigning multiple nets to the same track, which limits the possibility of achieving vertical lengths close to the total y-distance. The results of the gm-5 benchmarks are shown in Fig. 4.9.

Table 4.1: Results on two-pin nets.

Benchmark (#net)	Dens. Vert.	Dist. x	Dist. y (%)	y-length (w^y) (%) LE	BCA	SDG w/o 6	SDG
gt-1 (5)	3	1.5	1.4 (100)	2.9 (207)	1.8 (129)	1.8 (129)	1.8 (129)
gt-2 (10)	5	3.1	2.2 (100)	6.0 (273)	5.1 (232)	5.1 (232)	4.7 (214)
gt-3 (100)	53	32.4	29.6 (100)	79.5 (269)	39.9 (135)	39.9 (135)	35.0 (118)
gt-4 (500)	258	169.4	167.0 (100)	350.9 (210)	196.3 (118)	196.3 (118)	176.6 (108)
gt-5 (1000)	517	343.0	331.6 (100)	674.7 (203)	381.2 (115)	381.2 (115)	342.8 (103)
gt-6 (5000)	2474	1682.0	1676.1 (100)	3352.4 (200)	1963.7 (117)	1963.7 (117)	1727.9 (103)
gt-7 (10000)	4997	3365.6	3340.6 (100)	6844.5 (205)	3932.4 (118)	3932.4 (118)	3420.5 (102)
gt-8 (50000)	25102	16755.2	16734.0 (100)	33683.7 (201)	19751.9 (118)	19742.2 (118)	16981.0 (101)
gt-9 (100000)	49894	33302.8	33396.2 (100)	67716.8 (203)	39545.6 (118)	39444.1 (118)	33828.7 (101)

Table 4.2: Results on two-pin nets with post-processing

Benchmark (#net)	LE+PP			BCA+PP			SDG+PP		
	y-length (%)	Time [s]		y-length (%)	Time [s]		y-length (%)	Time [s]	
		LE	PP		BCA	PP		SDG	PP
gt-1 (5)	2.5 (180)	0.1	0.0	1.8 (129)	0.1	0.0	1.8 (129)	0.1	0.0
gt-2 (10)	4.7 (214)	0.1	0.0	3.7 (168)	0.1	0.0	3.7 (168)	0.1	0.0
gt-3 (100)	33.6 (114)	0.1	0.1	31.3 (106)	0.1	0.1	31.6 (107)	0.1	0.1
gt-4 (500)	176.1 (104)	0.1	0.6	170.4 (102)	0.2	0.2	170.3 (102)	0.2	0.3
gt-5 (1000)	347.0 (105)	0.2	1.6	335.0 (101)	0.2	0.8	335.2 (101)	0.2	1.0
gt-6 (5000)	1749.5 (104)	0.3	22.7	1688.5 (101)	1.5	11.3	1690.0 (101)	2.0	14.6
gt-7 (10000)	3483.6 (104)	0.5	87.3	3359.8 (101)	6.3	61.1	3359.0 (101)	8.7	41.2
gt-8 (50000)	17480.3 (104)	1.8	3374.3	16829.4 (101)	224.2	2174.7	16829.2 (101)	214.5	1447.7
gt-9 (100000)	34715.3 (104)	7.5	18990.9	33588.7 (101)	1283.1	12433.2	33588.3 (101)	1201.2	8256.9

Table 4.3: Results on multi-pin nets.

Benchmark (#net)	Density Vertical	Distance x	Distance y (%)	y-length (w^y) (%) LE	BCA	SDG w/o step 6	SDG
gm-1 (5)	5	3.1	4.4 (100)	7.2 (164)	6.8 (155)	6.6 (150)	6.6 (150)
gm-2 (10)	10	6.8	13.7 (100)	21.1 (154)	18.1 (132)	16.7 (122)	16.7 (122)
gm-3 (100)	89	65.7	128.9 (100)	203.7 (158)	179.4 (139)	151.8 (118)	149.9 (116)
gm-4 (500)	455	334.9	653.8 (100)	995.6 (152)	905.5 (138)	735.6 (113)	723.2 (111)
gm-5 (1000)	893	659.3	1260.8 (100)	1937.1 (154)	1709.3 (136)	1396.9 (111)	1377.4 (109)
gm-6 (5000)	4466	3315.8	6405.8 (100)	10077.0 (157)	8782.8 (137)	7194.4 (112)	7076.6 (110)
gm-7 (10000)	8901	6621.6	12652.7 (100)	19680.6 (156)	17292.2 (137)	14146.5 (112)	13891.3 (110)
gm-8 (50000)	44425	33094.3	63586.8 (100)	99454.0 (156)	86677.9 (136)	70838.7 (111)	69574.2 (109)
gm-9 (100000)	88832	66146.9	127098.3 (100)	198695.7 (156)	173344.5 (136)	141754.7 (111)	139113.5 (109)

Table 4.4: Results on multi-pin nets with post-processing

Benchmark (#net)	LE+PP				SDG+PP			
	y-length (%)	Time [s] (%)			y-length (%)	Time [s] (%)		
		LE	PP			SDG	PP	SDG+PP
gm-1 (5)	4.8 (109)	0.1 (97)	0.0	(42)	4.8 (109)	0.1 (89)	0.0 (11)	0.1 (100)
gm-2 (10)	15.9 (116)	0.1 (89)	0.0	(54)	15.9 (116)	0.1 (86)	0.0 (14)	0.1 (100)
gm-3 (100)	142.2 (110)	0.1 (50)	0.1	(49)	143.2 (111)	0.1 (56)	0.1 (44)	0.2 (100)
gm-4 (500)	703.6 (108)	0.2 (20)	0.6	(78)	704.4 (108)	0.2 (23)	0.6 (77)	0.8 (100)
gm-5 (1000)	1343.6 (107)	0.2 (7)	1.8	(77)	1341.0 (106)	0.3 (12)	2.1 (88)	2.4 (100)
gm-6 (5000)	6944.6 (108)	0.3 (1)	41.6	(98)	6906.2 (108)	3.4 (8)	39.1 (92)	42.5 (100)
gm-7 (10000)	13616.6 (108)	0.5 (0)	298.8	(110)	13554.8 (107)	14.7 (5)	257.1 (95)	271.8 (100)
gm-8 (50000)	68991.7 (108)	3.3 (0)	12889.2	(129)	68228.6 (107)	438.3 (4)	9553.3 (96)	9991.6 (100)
gm-9 (100000)	137774.6 (108)	13.8 (0)	73687.9	(130)	133834.5 (107)	2211.5 (4)	54471.5 (96)	56683.0 (100)

4.6 Summary

This chapter presented the development and application of a length minimization algorithm for the GCRP, aimed at reducing total vertical wire length to alleviate global congestion. The approach begins with the BCA algorithm, optimized for routing two-pin nets. The BCA priority system takes into account the relative positioning of a net's pins to the track, enhancing routing efficiency by minimizing wire length through strategic net prioritization.

For multi-pin nets, we introduced the SDG algorithm, a greedy heuristic that utilizes symmetric differences between pins to establish net priorities and manage track assignments effectively. Although SDG does not guarantee global optimality, it consistently outperforms traditional methods such as Left-Edge and BCA, leading to more compact net configurations in test scenarios.

A key advancement is the integration of post-processing techniques, which iteratively refine net assignments to further reduce vertical lengths. This process uses shifts and exchanges to reach a locally optimal solution, significantly enhancing solution quality. Experimental results show that SDG achieved a 50% reduction in vertical length compared to Left-Edge and 13% compared to BCA for two-pin nets. For multi-pin nets, the reductions were 30% and 20%, respectively.

Looking forward, while the current GCRP formulation provides a solid foundation, future work will expand on it to tackle the complexities of routing in critical layers and manage constraints like maximum congestion. SDG is expected to remain a key tool in advanced chip design, particularly for addressing the intricate routing challenges in critical layers.

Chapter 5

Local Congestion Alleviation Algorithm of GCRP

In GCRP, horizontal capacity is tight, requiring vertical wires to be routed on alternative layers. However, vertical routing resources are also limited, and routing must be completed within these constraints. If the demand for vertical wires exceeds the available vertical routing resources, routing becomes impossible, even if the total vertical wiring required for net connections fits within the overall available resources.

High local congestion exacerbates this issue, as it leads to bottlenecks that prevent the routing from being completed in the subsequent detailed routing phase. If local congestion is not managed effectively, the design will fail during detailed routing, even if it appears feasible at the global routing stage. Therefore, it is crucial to address local congestion at this stage to ensure that routing can be successfully completed in later phases.

To successfully achieve routing within these layers, local congestion must be kept below the threshold of available vertical routing resources. Thus, the primary objectives are to minimize the number of horizontal tracks utilized and reduce local congestion within these layers, ensuring a smooth transition to the detailed routing phase and preventing routing failure.

5.1 Definition of Local Congestion

In this chapter, both congestion and local congestion are used as metrics to evaluate track assignments. The congestion of a routing, resulting from a track assignment $\mathcal{A}$, is assessed by the total y-length as follows:

$$w^y(N_{\text{in}}, \mathcal{A}) = \sum_{n \in N_{\text{in}}} w^y(n, \mathcal{A}(n)) \tag{5.1}$$

Here, $w^y(n, \mathcal{A}(n))$ represents the y-length associated with a net n assigned to a track $\mathcal{A}(n)$, and N_{in} is the set of nets under consideration.

Local congestion is a more refined evaluation of congestion, focusing specifically on the common vertical segments between parallel wires of different nets that are in close proximity. If the vertical wires $v(p,t)$ and $v(p',t')$ belong to different nets and share a vertical section, and the horizontal distance between their respective pins is smaller than a predefined threshold d_{th}, we define this as causing local congestion. This is expressed as:

$$|x(p) - x(p')| \leq d_{\text{th}} \tag{5.2}$$

In this case, the vertical wires $v(p,t)$ and $v(p',t')$ are considered to contribute to local congestion by the overlapping parallel wire length. If this condition is not met, no local congestion is considered to occur between the two wires. The threshold d_{th} is a user-defined parameter that can be adjusted to suit the specific design constraints.

Let $P(N)$ represent the set of nearby pin pairs from the net set N, where the horizontal distance between pins is below the threshold d_{th}. This set is defined as:

$$P(N) = \{(p,p') \mid p \in n \in N, p' \in n' \in N, n \neq n', |x(p) - x(p')| \leq d_{\text{th}}, y(p) \geq y(p')\}.$$

Here, p and p' are pins from nets n and n', respectively, and their horizontal distance must satisfy $|x(p) - x(p')| \leq d_{\text{th}}$.

The *length of parallel wire* between two vertical wires $v(p,t)$ and $v(p',t')$, where $(p,p') \in P(N_{\text{in}})$, is defined based on their relative vertical positions as follows:

- Case 1: $y(t) > y(t')$:

$$PL\left(v(p,t), v(p',t')\right) = \begin{cases} y(t') - y(p) & \text{if } y(t') \geq y(p) \quad \text{(a)} \\ 0 & \text{if } y(p) > y(t') \wedge y(t) > y(p') \quad \text{(b)} \\ y(p') - y(t) & \text{if } y(p') \geq y(t) \quad \text{(c)} \end{cases}$$

- Case 2: $y(t') \geq y(t)$:

$$PL\left(v(p,t), v(p',t')\right) = \begin{cases} y(t) - y(p) & \text{if } y(t) \geq y(p) \quad \text{(d)} \\ \min(y(p), y(t')) - \max(y(p'), y(t)) & \text{if } y(p) \geq y(t) \wedge y(t') > y(p') \quad \text{(e)} \\ y(p') - y(t') & \text{if } y(p') \geq y(t') \quad \text{(f)} \end{cases}$$

Figure 5.1 illustrates six distinct cases based on the combinations of two vertical wires, with the corresponding parallel wire lengths (PL) shown for each scenario. The cases are categorized according to the relative positions of the vertical wires.

- **Cases with** $y(t) > y(t')$:
  1. (a) $y(t') \geq y(p)$, shown in Figure 5.1(a).
  2. (b) $y(p) > y(t') \wedge y(t) > y(p')$, shown in Figure 5.1(b).
  3. (c) $y(p') \geq y(t)$, shown in Figure 5.1(c).
- **Cases with** $y(t') \geq y(t)$:
  1. (d) $y(t) \geq y(p)$, shown in Figure 5.1(d).
  2. (e) $y(p) > y(t) \wedge y(t') > y(p')$, shown in Figure 5.1(e).
  3. (f) $y(p') \geq y(t')$, shown in Figure 5.1(f).

Each scenario corresponds to a different geometric relationship between the vertical wires, and the respective parallel wire lengths (PL) are determined by these configurations.

5.1.1 Total Parallel Length (TPL) Definition

Total Parallel Length (TPL) for a given set of nets, N_{in}, and a track assignment $\mathcal{A}$, is defined as the sum of the parallel wire lengths for each pair of nearby pins within the set of nets. Formally, the TPL is given by:

$$TPL(N_{\text{in}}, \mathcal{A}) = \sum_{(p,p') \in P(N_{\text{in}})} PL\left(v(p, \mathcal{A}(n(p))), v(p', \mathcal{A}(n(p')))\right)$$

where $P(N_{\text{in}})$ represents the set of pin pairs within proximity, and $PL\left(v(p,t), v(p',t')\right)$ denotes the parallel wire length between vertical wires $v(p,t)$ and $v(p',t')$, as previously described.

5.1.2 Conditions for No Parallel Wire

No parallel wire occurs between two vertical wires $v(p,t)$ and $v(p',t')$ under the following conditions:

- If $n(p) = n(p')$, meaning both pins belong to the same net.
- If the horizontal distance between the pins exceeds the threshold: $|x(p) - x(p')| > d_{\text{th}}$.
- If the vertical wires do not overlap according to the following conditions: $y(t) > y(t')$ and the corresponding inequalities for the pins p and p' are satisfied.

5.1.3 Pin Width Considerations

In the model described, pins are treated as points. However, in practical scenarios, pins may have physical widths, which could affect the calculation of parallel wire lengths. To accommodate this, the model can be adjusted to represent pins as intervals, with corresponding modifications to the algorithm to handle the effects of pin widths.

5.1.4 Objective of GCRP for Minimizing Total Local Congestion

The primary goal of the GCRP in this context is to minimize the total local congestion, as evaluated by the TPL, while assigning all nets in the set N_{in} to the available tracks in T_{in}.

The GCRP for minimizing the total local congestion is defined as follows:

GCRP for Minimizing Total Local Congestion

Instance: A set of nets N_{in} and a set of tracks T_{in}.

Output: A track assignment function $\mathcal{A} : N_{\text{in}} \to T_{\text{in}}$.

Objective: Minimize the total parallel length $TPL(N_{\text{in}}, \mathcal{A})$.

Constraints:
1. Each net $n \in N_{\text{in}}$ is assigned to a track $\mathcal{A}(n) \in T_{\text{in}}$.
2. The intervals $I(n_i)$ and $I(n_j)$ do not overlap if $\mathcal{A}(n_i) = \mathcal{A}(n_j)$, for all nets $n_i, n_j \in N_{\text{in}}$.

This problem is believed to be NP-hard, as there is evidence suggesting that a simple greedy algorithm cannot guarantee an optimal solution. However, to the best of our knowledge, this has not been formally proven yet.

While minimizing the TPL does not directly guarantee the elimination of actual local congestion (which depends on the actual routing results), it serves as a critical first step toward alleviating local congestion. By reducing the total parallel wire length, the likelihood of routing conflicts, and thus the occurrence of congestion, is minimized. This contributes to more efficient routing in the context of GCRP.

5.2 UEO Channel Routing Algorithm

The UEO (Under, Enclose, Over) routing algorithm [23], designed for generalized channels, aims to minimize the total parallel length (TPL) of vertical wires while using the

fewest horizontal tracks. Based on the GRFG framework, UEO ensures that trunk assignments for nets to tracks are completed when sufficient tracks are available.

The algorithm iteratively assigns trunks to tracks according to a prioritization scheme that targets CZ of each track under the GRFG framework. While UEO is a heuristic approach and does not guarantee optimality in terms of minimal TPL, it effectively reduces the TPL.

5.2.1 Relief Zone

The occurrence of parallel wires in vertical wire assignments depends on nearby pin pairs. Parallel wires emerge from the track assignments of net trunks, and proper assignments can prevent such occurrences.

To address this, we introduce the concept of the *relief zone* for a pin, which indicates tracks that minimize the likelihood of parallel wires. The relief zone for pin p, denoted as $R(p)$, is a range of y-coordinates. Ideally, the trunk of the net containing pin p is assigned to a track within $R(p)$. The relief zone for a net n, denoted as $R(n)$, is defined based on the relief zones of its pins.

If all net trunks are assigned within their relief zones, parallel wires will not occur. Our algorithm seeks to assign trunks within their relief zones to minimize parallel wire occurrence, though this may not always be feasible.

The relief zone for pin p is defined as:

$$R(p) = [R_{\min}(p), R_{\max}(p)],$$

where

$$R_{\min}(p) = \max\left(\max_{(p,p')\in P(N_{\text{in}})} y(p'), -\infty\right)$$

and

$$R_{\max}(p) = \begin{cases} y(p) & \text{if } \{p' \mid (p', p) \in P(N_{\text{in}})\} \text{ is empty,} \\ \infty & \text{otherwise.} \end{cases}$$

The relief zone of a pin represents the range of track assignments for that pin. To extend this concept to the net level, the relief zone of a net is defined as follows: If the intersection of the relief zones of the net's pins is non-empty, this intersection forms the relief zone for the net. If the intersection is empty, the relief zone for the net is defined by the relief zone of the pin whose upper boundary $R_{\max}(n)$ is the minimum among the relief zones of the net's pins. Thus, the relief zone for net n is:

$$R(n) = [R_{\min}(n), R_{\max}(n)].$$

Assigning a net trunk to a track within its relief zone does not guarantee minimal TPL, but it is expected to reduce parallel wire occurrences for at least one pin of the net.

In Fig. 5.1, the relief zones of pins p and p' are shown as $R(p) = [y(p'), \infty]$ (shaded blue) and $R(p') = [-\infty, y(p')]$ (shaded orange), respectively. In Fig. 5.1(b), the trunks of nets $n(p)$ and $n(p')$ are assigned to tracks t_3 and t_1, respectively. Here, $y(t_3) \in R(p)$ and $y(t_1) \in R(p')$, preventing parallel wires. In contrast, in Figs. 5.1(a), (d), and (e), the trunk of $n(p')$ is assigned to a track outside $R(p')$, causing parallel wires. Similarly, parallel wires occur in Figs. 5.1(c) and (f).

5.2.2 UEO Priority

In GRFG based algorithms [22], net trunks are assigned according to a specified net priority. The UEO (Under, Enclose, Over) net priority aims to minimize the total parallel length (TPL) by assigning trunks sequentially from the bottom track to the top track.

Nets are classified into three categories based on the relative position of their relief zone to the assigned track: type-U, type-E, and type-O. The priority order is as follows: type-U nets have the highest priority, followed by type-E nets, and finally type-O nets.

Additionally, within each type, nets are prioritized to further reduce TPL and total vertical wire length.

The classification of a net n for track t is determined as follows:

1. Type-U: $R_{\max}(n) \leq y(t)$,
2. Type-E: $R_{\max}(n) > y(t) \geq R_{\min}(n)$,
3. Type-O: $R_{\min}(n) > y(t)$ (i.e., the net's relief zone is entirely below the track).

Type-U nets have the highest priority because they are located above the current track. Assigning these nets to tracks above their relief zone prevents parallel wires, thus minimizing parallel length. Within type-U nets, priority is given based on the number of pins whose relief zones lie below track t, denoted as:

$$\texttt{RB}(n,t) = |\{p \in n \mid R_{\max}(p) < y(t)\}|.$$

In case of ties, the symmetric difference (SD) is used to measure the difference between the number of pins above and below the track [22], defined as:

$$\texttt{SD}(n,t) = |\{p \in n \mid y(p) < y(t)\}| - |\{p \in n \mid y(p) > y(t)\}|.$$

If ties persist, the leftmost position principle is applied. Type-E nets are given second priority. While assigning their trunk to the current track can help avoid parallel wires, they can still be placed on tracks outside their relief zone without causing parallel wire occurrences.

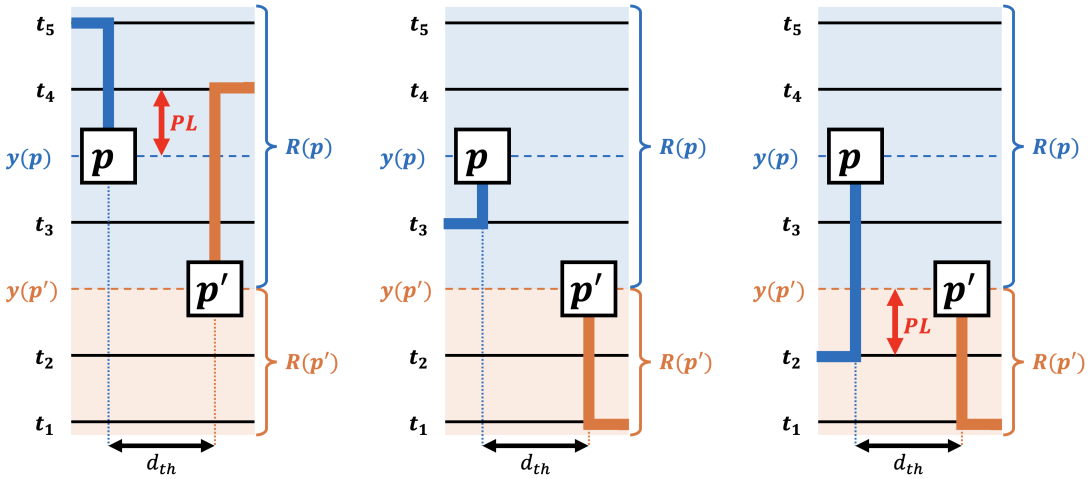
For type-E nets, ties are resolved using the SD-sequence, followed by the leftmost position principle.

Type-O nets receive the lowest priority. Since their relief zone is below the current track, placing their trunk there would cause parallel wires. However, if the trunk is not assigned to the current track, it can still be placed within the relief zone.

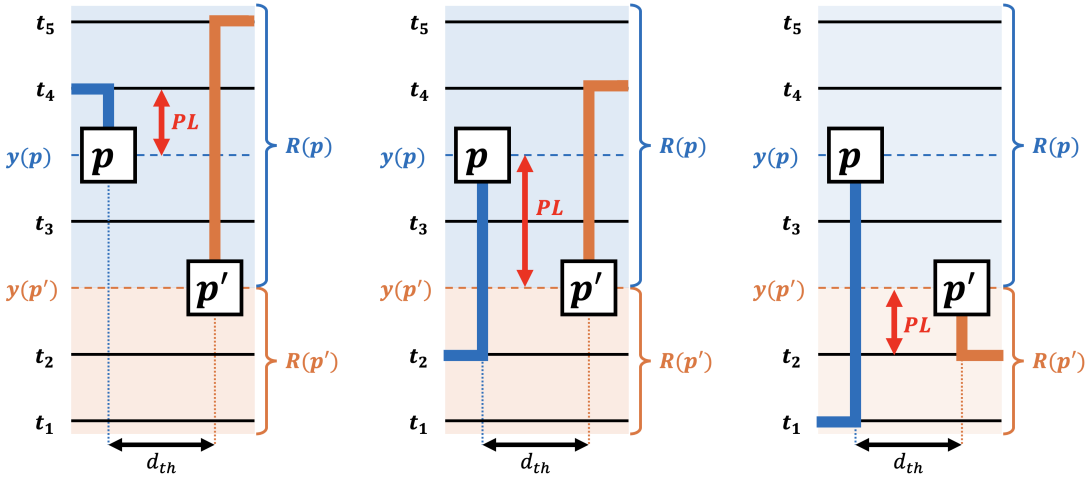
For type-O nets, priority is assigned based on the number of pins whose relief zones lie above track t, expressed as:

$$\texttt{RA}(n,t) = |\{p \in n \mid R_{\min}(p) > y(t)\}|.$$

As with the other types, ties are resolved first by the SD-sequence, then by the leftmost position principle.

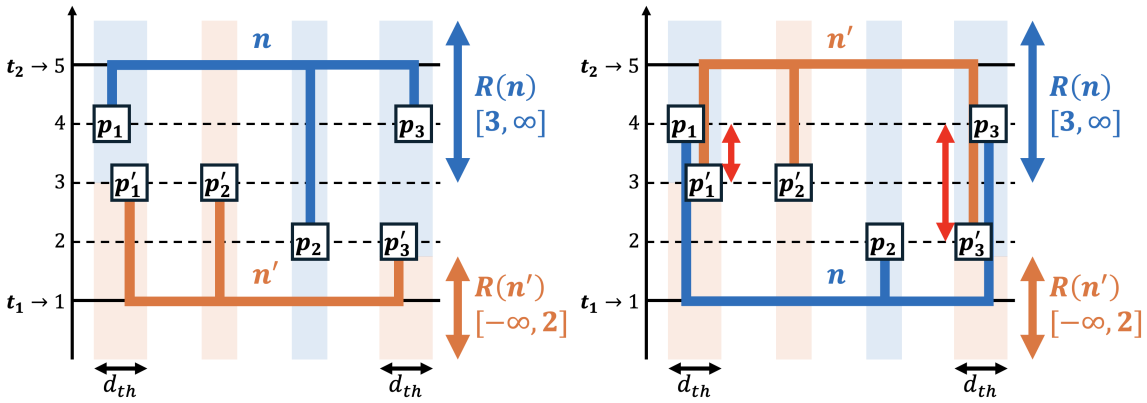


(a) $y(t') \geq y(p)$ (b) $y(p) > y(t') \wedge y(t) > y(p')$ (c) $y(p') \geq y(t)$



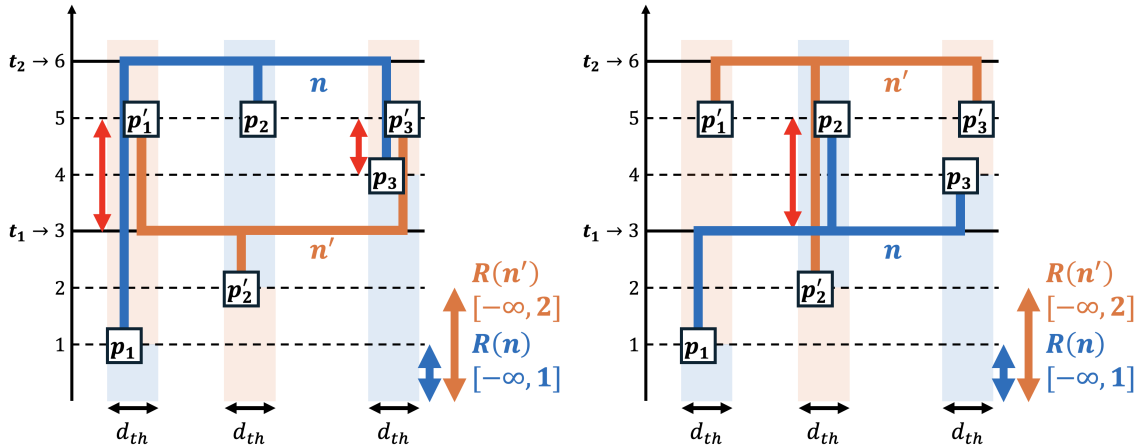
(d) $y(t) \geq y(p)$ (e) $y(p) \geq y(t) \wedge y(t') > y(p')$ (f) $y(p') \geq y(t')$

Figure 5.1: The length of parallel wire (PL).



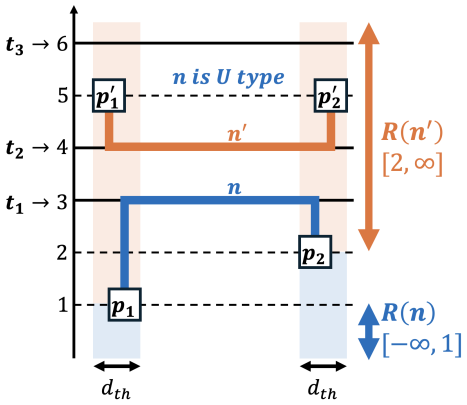
(a) Within the relief zone (b) Outside the relief-zone

Figure 5.2: Relief-zone and track assignments

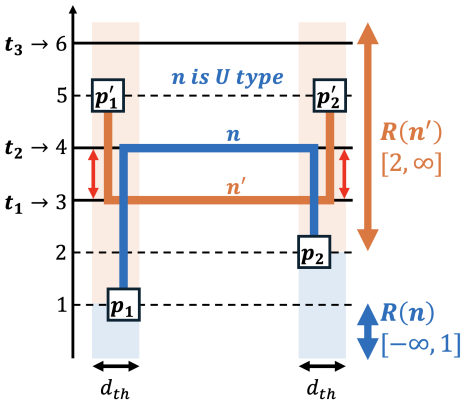


(a) Two parallel wires (b) One parallel wire

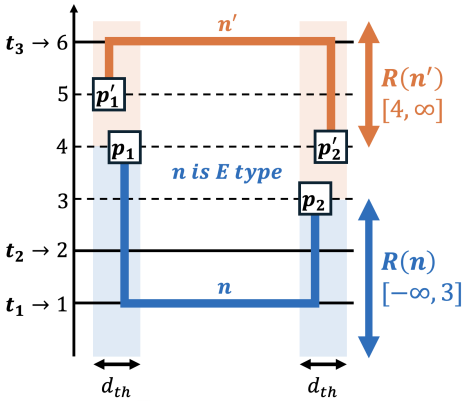
Figure 5.3: Relief-zone and track assignments



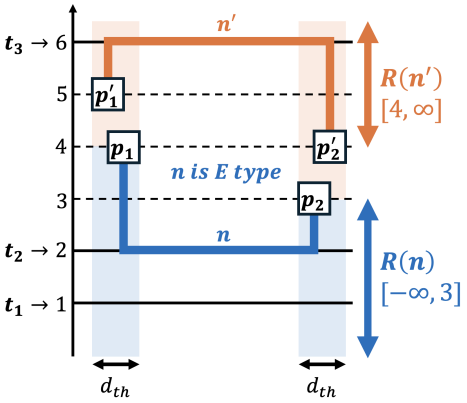
(a) Type-U assigns below
(#PW = 0, TPL = 0)



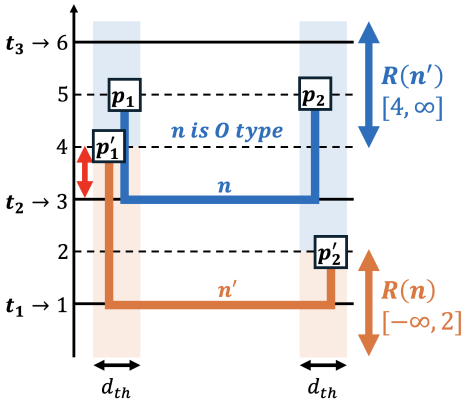
(b) Type-U assigns above
(#PW = 2, TPL = 2)



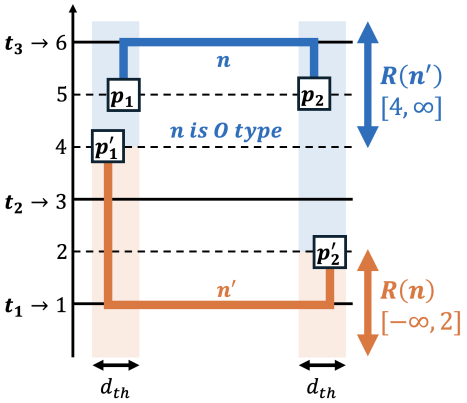
(c) Type-E assigns below
(#PW = 0, TPL = 0)



(d) Type-E assigns below
(#PW = 0, TPL = 0)

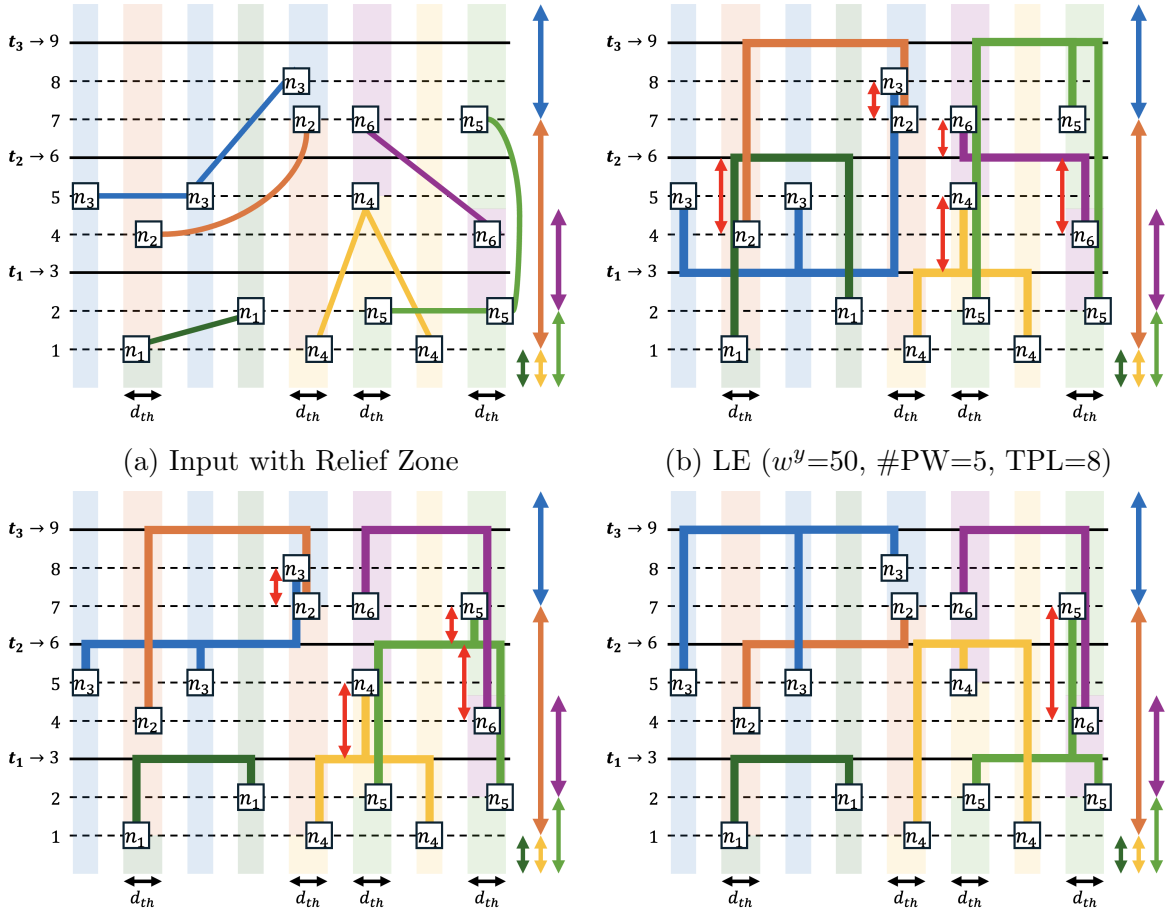


(e) Type-O assigns below
(#PW = 1, TPL = 1)



(f) Type-O assigns above
(#PW = 0, TPL = 0)

Figure 5.4: Three Types of Net



(a) Input with Relief Zone

(b) LE (w^y=50, #PW=5, TPL=8)

(c) SDG (w^y=36, #PW=4, TPL=6)

(d) UEO (w^y=39, #PW=1, TPL=3)

Figure 5.5: Track assignments

In Figs. 5.4(a) and (b), the relief zones of nets n and n' are $R(n) = [-\infty, 1]$ (blue arrow) and $R(n') = [2, \infty]$ (orange arrow), respectively. Net n is classified as type-U, and net n' as type-O with respect to tracks t_1 and t_2.

In Fig. 5.4(a), the trunks of n and n' are assigned to lower track t_1 and upper track t_2, respectively, resulting in no parallel wires (#PW=0) and a total parallel length (TPL) of 0. In Fig. 5.4(b), the trunks of n and n' are swapped to upper track t_2 and lower track t_1, leading to two parallel wires (#PW=2) and a TPL of 2. This suggests that assigning the trunk of a type-U net to a lower track is advantageous.

In Figs. 5.4(c) and (d), the relief zones of n and n' are $R(n) = [-\infty, 3]$ and $R(n') = [4, \infty]$, respectively. Net n is type-E, and n' is type-O with respect to tracks t_1 and t_2. In both Figs. 5.4(c) and (d), the trunks of n and n' are assigned to tracks

Algorithm 5 UEO Channel Routing Algorithm

Require: set of nets N_{in}, set of tracks $T_{\text{in}} = \{t_1, t_2, \ldots, t_m\}$ where $y(t_1) < y(t_2) < \ldots < y(t_m)$

Ensure: track assignment $\mathcal{A}(n)$ for all $n \in N_{\text{in}}$

$i \leftarrow 0$, $N \leftarrow N_{\text{in}}$, $T \leftarrow T_{\text{in}}$
while $N \neq \emptyset$ **do**
    $T \leftarrow T \setminus \{t_i\}$, $i \leftarrow i + 1$, $x \leftarrow -\infty$, $U \leftarrow N$
    while $U \neq \emptyset$ **do**
        $n \leftarrow$ Top_UEO_Priority(U, t_i), $U \leftarrow U \setminus \{n\}$
        if $(x, \infty) \cap Z(T, N) = \emptyset$ **and** $R_{\min}(n) > y(t)$ **then**
            break
        end if
        if $x \leq x_{\min}(n)$ **and** $(x, x_{\min}(n)) \cap Z(T, N) = \emptyset$ **then**
            $\mathcal{A}(n) \leftarrow t_i$, $x \leftarrow x_{\max}(n)$, $N \leftarrow N \setminus \{n\}$, $U \leftarrow N$
        end if
    end while
end while

within their respective relief zones, resulting in no parallel wires. This demonstrates that even when the trunk is not assigned to a lower track, parallel wires can still be avoided.

In Figs. 5.4(e) and (f), the relief zones for n and n' are $R(n) = [4, \infty]$ and $R(n') = [-\infty, 2]$, respectively. Net n is type-O, and n' is type-E relative to tracks t_2 and t_1. In Fig. 5.4(e), the trunks of n and n' are assigned to upper track t_2 and lower track t_1, resulting in parallel wires. However, in Fig. 5.4(f), when the trunks are assigned to upper track t_3 and lower track t_1, no parallel wires occur. This indicates that assigning the trunk of a type-O net to a higher track is more favorable.

In Fig. 5.5, the relief zones of six nets n_1, n_2, n_3, n_4, n_5, and n_6 are depicted, with each net's relief zone highlighted by colored arrows. The UEO net priority for track t_1 is given as $(n_5, n_1, n_4, n_2, n_6, n_3)$, where the nets are classified into three categories: type-U, type-E, and type-O. Specifically, nets n_1, n_4, and n_5 are classified as type-U, nets n_2 and n_6 as type-E, and n_3 as type-O.

Among the type-U nets, net n_5 holds the highest priority due to its superior relief zone representation, characterized by $\texttt{RB}(n_5, t_1) = 2$, which is greater than the relief zone

values of nets n_1 and n_4, both of which have $\texttt{RB}(n_1, t_1) = \texttt{RB}(n_4, t_1) = 1$. This higher relief zone representation of n_5 reflects its more favorable positioning relative to track t_1, making it the first to be assigned.

Within the type-U nets, the priority between n_1 and n_4 is determined by their SD value with respect to track t_1. Net n_1 is ranked higher than n_4 because its SD value $\texttt{SD}(n_1, t_1) = 2$ exceeds $\texttt{SD}(n_4, t_1) = 1$, indicating that n_1 is less constrained and more adaptable in its track assignment.

Among the type-E nets, net n_2 ranks higher than n_6 despite both nets having the same SD value $\texttt{SD}(n_2, t_1) = \texttt{SD}(n_6, t_1) = -2$. This is because the minimal x-coordinate of n_2, represented by $x_{\min}(n_2)$, is smaller than that of n_6, i.e., $x_{\min}(n_2) < x_{\min}(n_6)$, giving it a higher priority in the track assignment.

The UEO net priorities for tracks t_2 and t_3 are given as $(n_4, n_5, n_1, n_6, n_2, n_3)$ and $(n_4, n_5, n_1, n_2, n_6, n_3)$, respectively. For track t_2, net n_4 takes precedence due to its favorable relief zone, followed by n_5, n_1, and so on. Similarly, for track t_3, the priority order is adjusted based on the relative positioning of nets in their respective relief zones, with n_4 and n_5 again at the top, and n_3 being the last to be assigned.

This procedure ensures that nets are assigned in a way that minimizes congestion and optimizes the overall routing efficiency by respecting the priority of each net according to its relief zone characteristics and SD value.

5.2.3 UEO Channel Routing Algorithm

The UEO channel routing algorithm is designed to assign the trunks of nets to tracks within the framework of GRFG. The primary objective of this algorithm is to minimize the total parallel length (TPL) by leveraging UEO net priority, thereby enhancing the efficiency of the routing process.

Given a set of nets N_{in} and a set of tracks T_{in}, the UEO algorithm determines the priority of each net for every track in T_{in}. The algorithm begins by invoking the function `Top_UEO_Priority` in Step 5, which selects the highest-priority unassigned net for a particular track t_i. This step ensures that the most critical nets are assigned first, allowing for a more efficient and optimized routing process.

The assignment strategy within the UEO framework is driven by the classification of nets into three types: type-U, type-E, and type-O. Type-U and type-E nets are prioritized for track assignment based on their UEO net priority. Since earlier assignments tend to reduce the total parallel length by preventing unnecessary overlaps, type-U and type-E nets are assigned first. These assignments help minimize congestion and facilitate more efficient use of available tracks.

In contrast, type-O nets are treated differently. These nets are not assigned immediately but are instead carefully managed to reduce the likelihood of creating parallel wires, which would increase the total parallel length. The presence of type-O nets is monitored throughout the routing process to ensure that their assignment does not negatively impact the overall routing quality by introducing excessive parallelism.

The net assignment process for any given track terminates in Step 6 when CZ of that track is fully occupied by previously assigned nets. At this point, no further assignments except for type-O nets can be made without violating HC. This constraint ensures that all nets are assigned in a way that preventing illegal overlaps and minimizing the risk of congestion or routing violations.

The UEO channel routing algorithm strikes a balance between efficiency and constraint satisfaction, making it a highly effective tool for global routing in complex grid-based systems. By prioritizing the assignment of type-U and type-E nets and carefully managing type-O nets, the algorithm optimizes the routing solution to minimize parallel wire lengths while maintaining the integrity of the routing constraints.

5.2.4 Time Complexity Analysis

The time complexity of the UEO algorithm is analyzed under the assumption that each net has a constant number of pins and that $D(N_{\text{in}}) = |T_{\text{in}}|$, where m represents the number of nets.

Step 1 of the algorithm is executed once, leading to a time complexity of $\mathcal{O}(m)$. In this step, N is initialized as N_{in}, and nets are iteratively removed from N while Steps 2 through 13 are executed until N is empty.

The time complexity for executing each step once (excluding Step 5) is $\mathcal{O}(m)$. Step 5, which selects the top-priority net, requires sorting the nets based on their priority.

Since the priority values for each net can be computed in $\mathcal{O}(1)$, the sorting process takes $\mathcal{O}(m \log m)$. Sorting is performed for each set of nets N, resulting in $\mathcal{O}(m)$ executions of Step 5. Therefore, the total time complexity for Step 5 is $\mathcal{O}(m^2 \log m)$.

During each iteration of the while loop, encompassing Steps 4 through 12, at least one net is assigned to a track, ensuring that the number of iterations is $\mathcal{O}(m^2)$. Consequently, the total time complexity for Steps 2 through 13 (excluding Step 5) is $\mathcal{O}(m^2) \times \mathcal{O}(m) = \mathcal{O}(m^3)$.

Combining these results, the overall time complexity of the UEO algorithm is $\mathcal{O}(m^3)$.

The effectiveness of the UEO net priority is demonstrated in Fig. 5.5(d), where the total parallel length (TPL) is reduced to 3 using the UEO algorithm. This is a significant improvement compared to the Left-Edge algorithm (TPL = 8) and the SDG algorithm (TPL = 6), highlighting the efficiency of the UEO net priority in minimizing TPL.

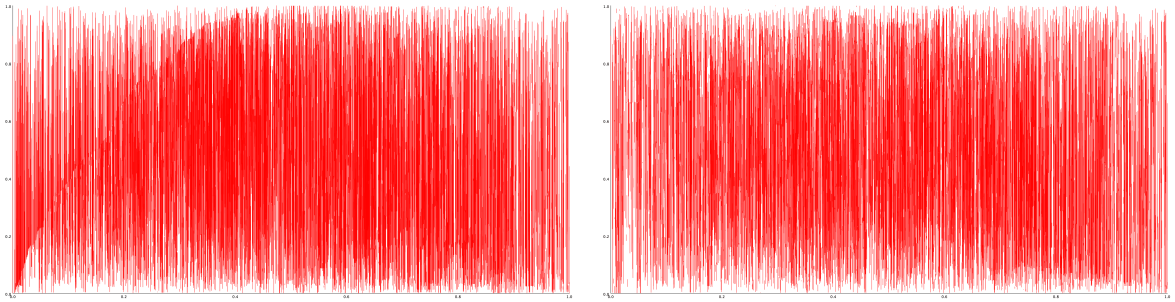
5.3 Experimental Results

5.3.1 Experimental Setup

The UEO routing algorithm, along with the Left-Edge [5] and SDG [22] algorithms for comparison, was implemented in Java using OpenJDK 22.0.2. All experiments were conducted on macOS, utilizing a 10-core Apple M1 Pro CPU with 16GB of memory. Each algorithm ran on a single CPU thread throughout the tests. The largest benchmark, consisting of 10,000 nets, completed in approximately 10 seconds.

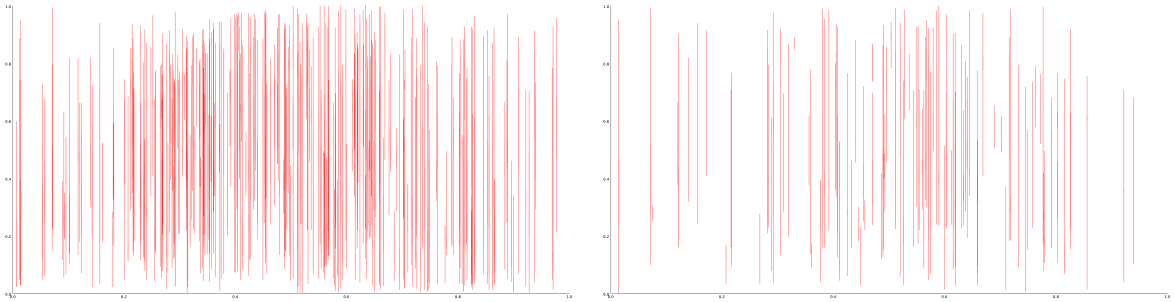
The benchmark evaluation includes eight groups with varying net counts, all labeled with the prefix "gm-" to indicate multi-pin configurations. Each group contains 10 instances. The pins are uniformly distributed with $x(p), y(p) \sim \mathcal{U}(0, 1)$ for all pins p in net n. The number of pins per net is randomly distributed between 2 and 10. The number of tracks in T_{in} equals $D(N_{\text{in}})$, and the y-coordinates of the tracks are uniformly distributed within the same range, i.e., $\sim \mathcal{U}(0, 1)$. The threshold d_{th} for the benchmark is set as $d_{\text{th}} = 0.012/\#\text{nets}$.

Table 5.1 presents the total y-lengths, while Table 5.2 displays the total parallel lengths (TPL) and average computation times.



(a) Left-Edge (w^y:2,045.1) (b) UEO (w^y:1,653.3)

Figure 5.6: Vertical wires in gm-6-0 (#net:1,000).



(a) Left-Edge (#PW:319, TPL:63.8) (b) UEO (#PW:120, TPL:9.3)

Figure 5.7: Parallel wires in gm-6-0 (#net:1,000).

In these tables, "LE," "SDG," and "UEO" represent the Left-Edge [5], SDG [22], and the proposed UEO algorithm, respectively.

The "Total Distance" is the sum of the total x-distance, $\sum_{n \in N_{\text{in}}} d^x(n)$, and the total y-distance, $\sum_{n \in N_{\text{in}}} d^y(n)$, where $d^x(n) = x_{\max}(n) - x_{\min}(n)$ and $d^y(n) = \sum_{p \in n} |y(p) - y(p_{\text{mid}})|$. Here, p_{mid} is the pin with the $\lceil |n|/2 \rceil$-th largest y-coordinate among pins in net n.

The "Ratio to Total y-Distance" refers to the ratio of the total y-length of the nets to their total y-distance. The "Total Parallel Length (TPL)" refers to the cumulative parallel lengths of the nets, using the average values from "LE" as a reference. TPL serves as a metric to assess local congestion in this research. For each group instance, three values are provided: average (ave.), minimum (min), and maximum (max).

5.3.2 Comparison with Conventional Methods

The results in Table 5.1 show that the UEO algorithm reduces the total y-length (w^y) by approximately 20% compared to the Left-Edge algorithm, but increases it by 16% relative to SDG. Table 5.2 further demonstrates that UEO reduces the total parallel length (TPL) by 85% compared to Left-Edge and 68% compared to SDG. However, the computation times for both SDG and UEO are longer than for Left-Edge.

The results from Tables 5.1 and 5.2 are visually summarized in Fig. 5.8(a), (b), and (c), where it is evident that the variability in total y-length, TPL, and computation time decreases as the number of nets increases.

Figures 5.6 and 5.7 compare the vertical wires for the "gm-6-seed-0" benchmark (1,000 nets) generated by Left-Edge and UEO, shown as red lines.

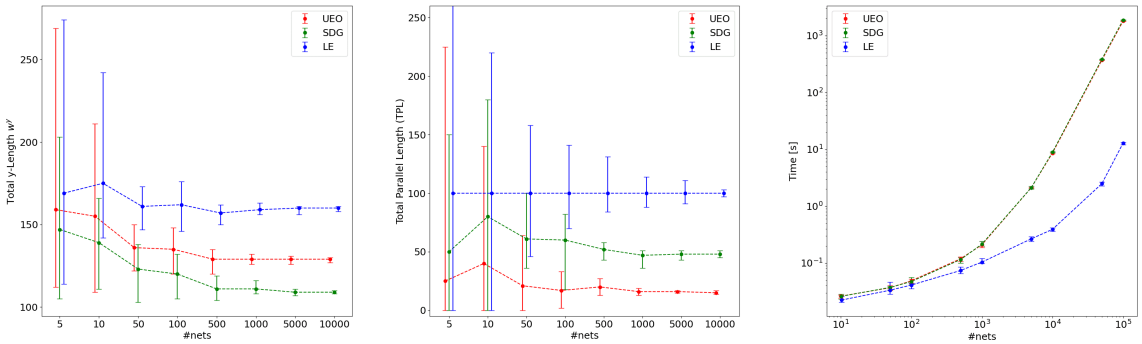
In Figs. 5.6(a) and (b), UEO results in a shorter vertical wire length ($w^y = 1,653.3$) compared to Left-Edge ($w^y = 2,045.1$).

Additionally, Figs. 5.7(a) and (b) display the parallel vertical wires. Left-Edge has 319 parallel wires, while UEO has only 120. UEO also achieves a significantly reduced total parallel length (TPL) of 9.1, compared to Left-Edge's 63.8. These results confirm UEO's effectiveness in reducing both the number of parallel wires and TPL.

5.3.3 Time Complexity Analysis

Figure 5.8(c) illustrates the computation times, which are typically under one second for smaller problem sizes, due to low-order terms or measurement limitations. This includes cases with $m = 50,000$ and $m = 100,000$. However, as the problem size increases, the computation times for larger instances become more pronounced. Specifically, for $m = 100,000$, the computation times for Left-Edge, SDG, and UEO are approximately 13 seconds, 1,900 seconds, and 1,800 seconds, respectively.

To estimate the time complexity of each algorithm, we analyze the differences in average computation times between $m = 50,000$ and $m = 100,000$. This approach yields the following time complexity estimates: - Left-Edge: $O(m^{2.37})$ - SDG: $O(m^{2.29})$ - UEO: $O(m^{2.29})$



(a) Total y-Length (w^y) (b) Total Parallel Length (TPL) (c) Computation time

Figure 5.8: Experimental results.

These results indicate that the UEO and SDG algorithms exhibit similar time complexities, with a polynomial growth rate of approximately $O(m^{2.29})$. In contrast, the Left-Edge algorithm has a slightly higher polynomial growth rate at $O(m^{2.37})$. Despite the similar complexities between UEO and SDG, UEO demonstrates better performance in terms of computation time for large-scale instances, with its execution time being notably shorter than SDG's.

In conclusion, while UEO and SDG have comparable time complexities, UEO offers a more efficient routing solution, especially for larger problem sizes. These findings emphasize UEO's potential for scalability in handling more complex routing instances, making it a viable option for large-scale VLSI designs.

Table 5.1: Comparison of Total y-Length (w^y).

Benchmark (#net)	Total Distance x	Total Distance y (%)	Ratio to Total y-Distance: ave.(%) (min(%), max(%)) LE	SDG	UEO
gm-1 (5)	1.5	5.8 (100)	169 (114, 274)	147 (105, 203)	159 (112, 269)
gm-2 (10)	3.2	11.4 (100)	175 (142, 242)	139 (111, 166)	155 (109, 211)
gm-3 (50)	17.5	63.7 (100)	161 (147, 173)	123 (103, 138)	136 (122, 150)
gm-4 (100)	34.0	127.5 (100)	162 (146, 176)	120 (105, 132)	135 (120, 148)
gm-5 (500)	166.6	642.8 (100)	157 (150, 162)	111 (104, 119)	129 (120, 135)
gm-6 (1,000)	333.5	1,285.7 (100)	159 (156, 163)	111 (108, 116)	129 (126, 132)
gm-7 (5,000)	1,665.1	6,390.4 (100)	160 (156, 161)	109 (107, 111)	129 (126, 131)
gm-8 (10,000)	3,331.2	12,765.9 (100)	160 (158, 161)	109 (108, 110)	129 (127, 130)

Table 5.2: Comparisons of Total Parallel Length (TPL) and Computation Time

Benchmark (#net)	TPL ave. (%) LE	Ratio to TPL ave. by LE: ave.(%) (min(%), max(%)) LE	SDG	UEO	Average Computation Time [s] LE	SDG	UEO
gm-1 (5)	0.4 (100)	100 (0, 360)	50 (0, 150)	25 (0, 225)	0.02	0.03	0.02
gm-2 (10)	0.5 (100)	100 (0, 220)	80 (0, 180)	40 (0, 140)	0.02	0.03	0.03
gm-3 (50)	2.8 (100)	100 (46, 158)	61 (36, 100)	21 (0, 64)	0.03	0.04	0.04
gm-4 (100)	6.0 (100)	100 (70, 141)	60 (18, 82)	17 (2, 33)	0.04	0.05	0.05
gm-5 (500)	30.6 (100)	100 (84, 131)	52 (43, 58)	20 (13, 27)	0.07	0.12	0.12
gm-6 (1,000)	67.3 (100)	100 (88, 114)	47 (36, 51)	16 (13, 19)	0.10	0.21	0.21
gm-7 (5,000)	319.0 (100)	100 (91, 111)	48 (43, 51)	16 (15, 17)	0.26	2.11	2.11
gm-8 (10,000)	634.8 (100)	100 (97, 103)	48 (45, 51)	15 (14, 17)	0.39	8.90	8.59

5.4 Summary

This chapter introduces the UEO (Under, Enclose, Over) routing algorithm, specifically designed to address the unique challenges posed by routing layers with tight horizontal capacity, commonly modeled as generalized channels. These layers present specific constraints, particularly regarding the limited availability of horizontal tracks, which necessitate efficient routing strategies to minimize congestion and optimize overall wire length. The UEO algorithm provides an efficient routing solution by connecting nets through a single-trunk Steiner tree structure, which is carefully tailored to meet these challenges while maintaining minimal wire length. The algorithm's design accounts for both the spatial constraints and the need to ensure that nets are routed in a manner that minimizes interference and congestion within the channel.

Experimental results demonstrate the efficacy of the UEO algorithm in reducing the total parallel length (TPL), a critical metric for evaluating local congestion in routing problems. Specifically, UEO achieves a reduction of approximately 85% in TPL compared to the Left-Edge algorithm and around 68% when compared to the SDG algorithm. These results highlight UEO's superior performance in managing routing efficiency, especially within environments where horizontal capacity is constrained. By significantly minimizing TPL, UEO not only enhances the quality of the routing solution but also reduces the potential for local congestion, a critical factor in ensuring the performance and manufacturability of integrated circuits.

Despite the substantial improvements achieved by UEO, there remains room for further optimization. In particular, while UEO effectively reduces congestion and wire length, it still has opportunities to improve in terms of addressing actual local congestion issues. Local congestion occurs when multiple nets share the same routing resources, potentially leading to interference and inefficiencies in the final layout. To further improve its performance, UEO could be enhanced to better handle these situations by incorporating dynamic routing strategies that adapt to changing congestion levels during the routing process. Moreover, refining the algorithm to be more flexible and adaptable to a broader range of design constraints, such as varying track densities, technology-specific restrictions, and noise considerations, could further extend its applicability to real-world VLSI design challenges.

Potential improvements to UEO could also involve integrating additional routing

strategies, such as multi-path routing or adaptive track assignment techniques, which may allow the algorithm to more effectively address complex scenarios where multiple routing layers and varying degrees of congestion must be managed simultaneously. Further optimization could also focus on improving the algorithm's scalability to handle larger, more complex routing tasks, as well as reducing its computational overhead in high-density environments.

In conclusion, the UEO algorithm demonstrates significant promise as a foundational tool for advanced chip designs, particularly in the context of emerging three-dimensional (3D) integration techniques in VLSI systems. As the demand for more compact and efficient circuit layouts grows with advances in chip manufacturing technologies, UEO's ability to efficiently manage complex routing scenarios and minimize congestion will make it a valuable asset for the ongoing evolution of VLSI technology. Its application in 3D IC design, where the need for efficient routing is amplified by the increased number of layers, positions it as a key technology in the development of next-generation chip designs. Ultimately, the advancements enabled by UEO could pave the way for more efficient, reliable, and compact circuit layouts, driving innovation in VLSI systems and contributing to the continued progress of semiconductor technology.

Chapter 6

Conclusion

This research addresses the challenges associated with routing design in three-dimensional (3D) integrated circuits, specifically focusing on the critical routing layers of advanced chips. The proposed strategies significantly enhance routing efficiency and effectiveness, particularly for high-performance and low-cost applications.

To tackle the challenges in critical routing layers, this research models the routing problem in generalized channels, which we refer to as the Generalized Channel Routing Problem (GCRP). Due to the tight horizontal capacity inherent in GCRP, a versatile routing framework, the Greedy Routing Framework with Guarantee (GRFG), is introduced. GRFG ensures the minimization of track usage while achieving various objectives such as minimizing total wire length and reducing local congestion. To further minimize vertical wire length in GCRP, the BCA channel routing algorithm is proposed for two-pin nets, followed by the Symmetric Difference General (SDG) channel routing algorithm for multi-pin nets. The SDG algorithm reduces vertical wire length by approximately 30%-50% compared to the Left-Edge algorithm. Additionally, the UEO channel routing algorithm is introduced to alleviate local congestion, achieving an 85% reduction in local congestion compared to the Left-Edge algorithm and a 68% reduction compared to SDG.

The methods and strategies proposed in this thesis have broad applications in 3D IC design, paving the way for more efficient routing algorithms and design automation tools. These strategies contribute to the advancement of 3D IC implementation and have significant implications for both engineering and industrial applications.

Future work will explore additional constraints and refine these methods to improve their applicability and effectiveness in practical design scenarios, further enhancing their role in modern chip design.

Acknowledgements

I would like to express my heartfelt gratitude to Professor Atsushi Takahashi for his invaluable guidance, unwavering support, and constant encouragement throughout my Ph.D. journey. His patience, expertise, and insightful advice have been essential to my perseverance and success in completing this dissertation. I am deeply thankful for his mentorship, which has profoundly shaped both my academic and personal growth. I would also like to extend my sincere thanks to Mr. Masayuki Shimoda for his valuable insights, constructive feedback, and steadfast encouragement. His contributions were pivotal in refining my research and helping me navigate the challenges of my work.

Furthermore, I owe a profound debt of gratitude to my parents and my wife for their unending love, support, and encouragement. My parents have always been a source of unwavering belief and inspiration, providing me with the foundation I needed to persevere through the toughest times. To my wife, whose patience, understanding, and sacrifices have been immeasurable, I am deeply thankful. Her constant support and love have been my greatest strength, and my Ph.D. journey is as much a reflection of her unwavering belief in me.

Publications

Journal Papers

[1] Wang, Zezhong, Shimoda, Masayuki, and Takahashi, Atsushi. "SDG Channel Routing to Minimize Wirelength for Generalized Channel." 2025, IEICE Transactions. Fundamentals. Vol.E108-A, No.3, pp.-, Mar. 2025. (to appear)

[2] Wang, Zezhong, Shimoda, Masayuki, and Takahashi, Atsushi. "UEO Channel Routing Algorithm to Alleviate Local Congestion for Generalized Channels." 2025, IEICE Transactions. Fundamentals. Vol.-, No.-, pp.-, Mar. 2025. (Submitted)

Conference Papers

[1] Wang, Zezhong, Shimoda, Masayuki, and Takahashi, Atsushi. "BCA Channel Routing to Minimize Wirelength for Generalized Channel Problem." 2024 IEEE International Symposium on Circuits and Systems (ISCAS), 2024, doi = 10.1109/ISCAS58744.2024.10558428.

[2] Wang, Zezhong, Shimoda, Masayuki, and Takahashi, Atsushi. "Single Trunk Routing Problem for Generalized Channel." IEICE Technical Report (VLD2023-104), 2024, vol.123, no.390, page 30-35.

[3] Wang, Zezhong, Shimoda, Masayuki, and Takahashi, Atsushi. "Optimization Algorithms for Generalized Channel Routing Problem." TJCAS, 2024.

Presentations

[1] Wang, Zezhong, Shimoda, Masayuki, and Takahashi, Atsushi. "Single Trunk Routing Problem." 77th Systems and Algorithms Joint Seminar.

[2] Wang, Zezhong, Shimoda, Masayuki, and Takahashi, Atsushi. "Single Trunk Routing Problem for Generalized Channel." IEICE Technical Report (VLD2023-104), 2024.

Posters

[1] Wang, Zezhong, Shimoda, Masayuki, and Takahashi, Atsushi. "BCA Channel Routing to Minimize Wirelength for Generalized Channel Problem." 2024 IEEE International Symposium on Circuits and Systems (ISCAS), 2024, C2P-16.

[2] Wang, Zezhong, Shimoda, Masayuki, and Takahashi, Atsushi. "Optimization Algorithms for Generalized Channel Routing Problem." TJCAS, 2024, P1-17.

Bibliography

[1] Gracieli Posser et al. “Challenges and Approaches in VLSI Routing”. In: *Proc. International Symposium on Physical Design (ISPD)*. 2022, pp. 185–192. DOI: 10.1145/3505170.3511477.

[2] M. Sako et al. “A 1Tb 3b/Cell 3D-Flash Memory of more than 17Gb/mm2 bit density with 3.2Gbps interface and 205MB/s program throughput”. In: *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. 2023, pp. 1–2. DOI: 10.23919/VLSITechnologyandCir57934.2023.10185237.

[3] S. Kobayashi et al. “High Peformance 3D Flash Memory with 3.2Gbps Interface and 205MB/s Program Throughput based on CBA(CMOS Directly Bonded to Array) Technology”. In: *2023 International Electron Devices Meeting (IEDM)*. 2023, pp. 1–4. DOI: 10.1109/IEDM45741.2023.10413716.

[4] Masayoshi Tagami. “CMOS Directly Bonded to Array (CBA) Technology for Future 3D Flash Memory”. In: *2023 International Electron Devices Meeting (IEDM)*. 2023, pp. 1–4. DOI: 10.1109/IEDM45741.2023.10413718.

[5] Akihiro Hashimoto and James Stevens. “Wire Routing by Optimizing Channel Assignment within Large Apertures”. In: *Proc. 8th Design Automation Workshop (DAC)*. 1971, pp. 155–169. ISBN: 9781450374651. DOI: 10.1145/800158.85069.

[6] David N. Deutsch. “A dogleg channel router”. In: *Proc. the 13th Design Automation Conference (DAC)*. 1976, pp. 425–433. DOI: 10.1145/800146.804843.

[7] A. S. LaPaugh. “Algorithms for integrated cricuit layout: An analytic approach”. PhD thesis. USA: Massachusetts Institute of Technology, 1980.

[8] T. Yoshimura and E.S. Kuh. “Efficient Algorithms for Channel Routing”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 1.1 (1982), pp. 25–35. DOI: 10.1109/TCAD.1982.1269993.

[9] T.G. Szymanski. “Dogleg Channel Routing is NP-Complete”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 4.1 (1985), pp. 31–41. DOI: 10.1109/TCAD.1985.1270096.

[10] T.-T. Ho, S.S. Iyengar, and S.-Q. Zheng. “A general greedy channel routing algorithm”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 10.2 (1991), pp. 204–211. DOI: 10.1109/43.68407.

[11] Atsushi Takahashi and Hiroshi Murata. “Three-layer L-shaped channel routing algorithm”. In: *IPSJ Journal* 40.4 (1999). (in Japanese), pp. 1618–1625.

[12] Swagata Saha Sau et al. “A graph based algorithm to minimize total wire length in VLSI channel routing”. In: *2011 IEEE International Conference on Computer Science and Automation Engineering*. Vol. 3. 2011, pp. 61–65. DOI: 10.1109/CSAE.2011.5952634.

[13] Kazuya Taniguchi et al. “Bottleneck Channel Routing to Reduce the Area of Analog VLSI”. In: *Proc. the 24th Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI)*. 2022, pp. 26–31.

[14] Kazuya Taniguchi et al. “A Fast Three-layer Bottleneck Channel Track Assignment with Layout Constraints using ILP”. In: *Proc. the 25th Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI)*. 2024, pp. 50–55.

[15] Kazuya Taniguchi et al. “Two-layer Bottleneck Channel Track Assignment for Analog VLSI”. In: *IPSJ Trans. on System LSI Design Methodology* 17 (2024), pp. 67–76. DOI: 10.2197/ipsjtsldm.17.67.

[16] Kazuya Taniguchi et al. “A Fast Three-layer One-side Bottleneck Channel Routing with Layout Constraints using ILP”. In: *IEICE Trans. Fundamentals* 108 (2025). (to appear). DOI: 10.1587/transfun.2024VLP0002.

[17] Zezhong Wang, Masayuki Shimoda, and Atsushi Takahashi. “Single Trunk Routing Problem for Generalized Channel”. In: *IEICE Technical Report (VLD2023-104)* 123.390 (2024), pp. 30–35.

[18] M. Borah, R.M. Owens, and M.J. Irwin. “An edge-based heuristic for Steiner routing”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 13.12 (1994), pp. 1563–1568. DOI: 10.1109/43.331412.

[19] Hongyu Chen et al. “Refined Single Trunk Tree: A Rectilinear Steiner Tree Generator for Interconnect Prediction”. In: *Proc. International Workshop on System-Level Interconnect Prediction (SLIP)*. 2002, pp. 85–89. DOI: 10.1145/505348.505366.

[20] Sheng-En David Lin and Dae Hyun Kim. “Construction of All Rectilinear Steiner Minimum Trees on the Hanan Grid and Its Applications to VLSI Design”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39.6 (2020), pp. 1165–1176. DOI: 10.1109/TCAD.2019.2917896.

[21] Zezhong Wang, Masayuki Shimoda, and Atsushi Takahashi. “BCA Channel Routing to Minimize Wirelength for Generalized Channel Problem”. In: *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2024, pp. 1–5. DOI: 10.1109/ISCAS58744.2024.10558428.

[22] Zezhong Wang, Masayuki Shimoda, and Atsushi Takahashi. “SDG Channel Routing to Minimize Wirelength for Generalized Channel”. In: *IEICE Trans. Fundamentals* 108 (2025). (Accepted). DOI: 10.1587/transfun.2024VLP0001.

[23] Zezhong Wang, Masayuki Shimoda, and Atsushi Takahashi. “UEO Channel Routing Algorithm to Alleviate Local Congestion for Generalized Channels”. In: *IEICE Trans. Fundamentals* (2025). (Accepted).

[24] Masayuki Shimoda and Atsushi Takahashi. “Gridless Gap Channel Routing with Variable-width Wires”. In: *IEICE Trans. Fundamentals* 108 (2025). (Accepted). DOI: 10.1587/transfun.2024VLP0003.

[25] Masayuki Shimoda and Atsushi Takahashi. “Wirelength Minimization by Gap Swap-Flip in Gridless Gap Channel Routing”. In: *Proc. IEEE International SoC Design Conference (ISOCC)*. 2024, pp. 213–214. DOI: 10.1109/ISOCC62682.2024.10762381.

[26] Masayuki Shimoda and Atsushi Takahashi. “Gridless Gap Channel Routing to Minimize Wirelength”. In: *IPSJ Trans. on System LSI Design Methodology* 18 (2025). (Accepted).

[27] Man-Pan Wong, Wen-Hao Liu, and Ting-Chi Wang. “Negotiation-based track assignment considering local nets”. In: *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. 2016, pp. 378–383. DOI: 10.1109/ASPDAC.2016.7428041.

[28] Fan-Keng Sun et al. “A Multithreaded Initial Detailed Routing Algorithm Considering Global Routing Guides”. In: *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2018, pp. 1–7. DOI: 10.1145/3240765.3240777.

[29] L. McMurchie and C. Ebeling. “PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs”. In: *Third International ACM Symposium on Field-Programmable Gate Arrays*. 1995, pp. 111–117. DOI: 10.1109/FPGA.1995.242049.

[30] Jinan Lou, Shankar Krishnamoorthy, and Henry S Sheng. “Estimating routing congestion using probabilistic analysis”. In: *Proceedings of the 2001 international symposium on Physical design*. 2001, pp. 112–117.

[31] Hannah Szentimrey et al. “Machine learning for congestion management and routability prediction within FPGA placement”. In: *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 25.5 (2020), pp. 1–25.

[32] Tamás Terlaky, Anthony Vannelli, and Hu Zhang. “On routing in VLSI design and communication networks”. In: *Discrete applied mathematics* 156.11 (2008), pp. 2178–2194. DOI: https://doi.org/10.1016/j.dam.2008.01.014.

[33] Kuruva Lakshmanna et al. “Perimeter Degree Technique for the Reduction of Routing Congestion during Placement in Physical Design of VLSI Circuits”. In: *Complex.* 2022.11 (2022). DOI: 10.1155/2022/8658770.

[34] Prashant Saxena, Rupesh S Shelar, and Sachin Sapatnekar. *Routing Congestion in VLSI Circuits: Estimation and Optimization*. Springer Science & Business Media, 2007.