Towards Better Physical Layout Routability and Security

WANG, Fangzhou

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy

in

Computer Science and Engineering

The Chinese University of Hong Kong

August 2023

Thesis Assessment Committee

Professor YU Bei (Chair)

Professor YOUNG Fung Yu (Thesis Supervisor)

Professor SHAO Zili (Committee Member)

Professor CHEN Hung-Ming (External Examiner)

Abstract

Very large-scale integration (VLSI) physical design involves the planning of geometric layouts for integrated circuits (ICs) based on abstract logical descriptions. Over the past few decades, advancements in technology nodes and integration levels have led to ICs with improved power, performance, and area (PPA). However, these advancements have also introduced new challenges to the existing physical design flow. VLSI routing has become increasingly difficult due to the growing number of design rules and the presence of millions, or even billions, of circuit components. This complexity hinders the circuit design process and leads to longer design periods. Additionally, the prevalence of fabless manufacturing has raised significant security concerns regarding physical layouts.

In this thesis, we present enhanced methodologies to address the aforementioned challenges related to routability and security.

Routing is typically divided into two steps: global routing and detailed routing, which helps reduce complexity. To improve routability, we focus on two problems in global routing and detailed routing, respectively. First, we address the misalignment between the objectives of placement and global routing. Placement usually aims to optimize half-perimeter wire length (HPWL) and estimations of routability, such as area density and pin density. On the other hand, global routing aims to minimize the routed wire length and number of overflows. As modern VLSI designs can

involve over ten routing layers, each with different routing resources and varying design rules, the correlation between the two-dimensional (2D) HPWL and the three-dimensional (3D) routed wire length is weak. Additionally, satisfying target density does not guarantee a routable design. In contrast, reserving a larger margin for routability can lead to longer wire length and delay. This misalignment in objectives can result in a significant degradation of solution quality. To bridge the gap between placement and routing (P&R), we propose an efficient P&R co-optimization framework that effectively reduces the total wire length of an initial layout without introducing congestion issues. Second, we tackle the intricate and time-consuming detailed routing problem, which becomes even more challenging when complex pin shapes are involved. To enhance detailed routability, we introduce a novel pin access analysis framework that achieves design rule checking (DRC)-clean pin access schemes and is significantly faster than the current state-of-the-art approach. We integrate this framework into a detailed router, resulting in substantial improvements in detailed routing quality.

For security, as IC supply chains are now largely outsourced due to cost benefits, various third-party providers are involved in the manufacturing process. However, this outsourcing gives rise to several kinds of threats, including piracy of IC intellectual property and the insertion of hardware Trojans—malicious circuit modifications. Therefore, in this thesis, we propose a multiplexer-based logic-locking scheme along with a security-aware physical design flow to prevent layout-level Trojan insertion.

摘要

超大規模集成電路（VLSI）物理設計是指依據抽象的邏輯描述，對集成電路（IC）進行幾何版圖（Layout）的規劃。 在過去幾十年中，製程的進步使得IC的功耗、性能和面積得到了顯著的優化，同時也為現有的物理設計流程帶來了新的挑戰。 在更先進的製程中，設計規則變得更加多樣化，電路上的元件數量激增至百萬甚至上十億的規模，這大大增加了VLSI佈線（Routing）的難度。 此外，無廠半導體製造的普及也帶來了物理佈局上的安全性隱患。

在本論文中，為了應對在可佈線性（Routability）和安全性（Security）方面的上述挑戰，我們提出了更高效的解決方案。

佈線分為兩個步驟：全局佈線和詳細佈線。為提升可佈線性，我們分別專注於全局佈線和詳細佈線中相應的問題。 首先，我們關注佈局（Placement）和全局佈線目標的不一致。 佈局的目標是優化半周長線長（HPWL）和可佈線性的估計，如區域密度和引腳密度，而全局佈線旨在最小化實際線長和擁塞情況。 現代VLSI設計涉及超過十層具有不同佈線資源和設計規則的佈線層，因此相較於以前，二維HPWL和三維實際線長的相關性進一步降低。 此外，即使在佈局階段滿足目標密度，也無法保證之後能成功進行佈線。 相對的，保留更大的目標密度餘量雖然可以確保可佈線性，但也會引入更多的線長和延遲。 類似這樣的不一致性將大幅度降低最終版圖的質量。 為了縮小佈局和佈線之間的差異，我們提出一個高效的佈局佈線協同優化框架，在有效減少已有版圖的總線長的同時，不會引入擁塞問題。 其次，我們專注複雜且耗時的詳細佈線問題，特別是涉及到複雜引腳形狀時，這個問題變

得更具挑戰性。 為了提高詳細可佈線性，我們提出了一個新穎的引腳訪問分析框架，其中的算法不僅能生成符合設計規則的引腳訪問方案，並且比當前最先進的方法更加高效。該框架在整合到詳細佈線器後顯著提升了詳細佈線的質量。

至於安全性，由於成本效益，IC供應鏈現在主要是外包給各種第三方提供商進行製造。 然而，這種外包引發了多種安全性上的威脅，包括IC知識產權（IP）盜竊和硬件木馬的插入。 因此，在本論文中，我們提出了基於多路複用器的邏輯鎖定方案以及注重安全性的物理設計流程，以在物理版圖層面防止木馬的插入。

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Evangeline F.Y. Young, for leading me into the field of electronic design automation, engaging in countless inspiring discussions, and providing me with precious guidance, kind encouragement, and continuous support. I consider myself fortunate to have met her four years ago during the CUHK summer workshop.

I would also like to thank my other thesis committee members, Prof. Bei Yu, Prof. Zili Shao and Prof. Hung-Ming Chen, for their insightful comments and constructive suggestions.

Additionally, I want to express my gratitude to my internship mentors at Cadence: Dr. Wing-Kai Chow and Dr. Mehmet Can Yildiz, for sharing their industry experience and providing valuable career development advice. It was my first time being outside of Asia, and I feel fortunate to have met Dr. Gracieli Posser, Dr. Mateus Fogaça, Dr. Tiago Augusto Fontana, Arnav Mehrotra, Dr. Jun Zhou, Dr. Zi Wang, Dr. Yixiao Ding and Jianfeng Song at Cadence. I am thankful for their warm assistance during my stay in Austin, Texas.

I would like to express my sincere thanks to my fellow students at the Chinese University of Hong Kong: Dr. Peishan Tu, Dr. Gengjie Chen, Dr. Chak-Wa Pui, Dr. Huan Shi, Dr. Haocheng Li, Dr. Yuzhe Ma, Dr. Haoyu Yang, Wei Li, Dr. Jingsong Chen, Dr. Bentian Jiang, Dr. Jinwei Liu, Dr. Xiaopeng Zhang, Dr. Dan Zheng, Lixin Liu, Xinshi Zang, Shiju Lin, Tianji Liu, Bangqi Fu, Qijing Wang, Wing Ho Lau, Yang Sun and Qin Luo. Thanks to them, my Ph.D. journey has been a memorable and cherished experience.

Special thanks go to my dearest friends: Jiamin Li, Kun Yang and Lin Jin, for their companionship since the beginning of my undergraduate studies, and for

sharing both my joy and sorrow. In particular, I want express my gratitude to Jiamin Li for consistently being a supportive and caring friend.

Finally, this thesis would not have been possible without the support and sacrifices of my family. I would like to thank my parents, Xiaoling Fang and Haibo Wang, for their endless love, encouragement, and their unwavering support, even during challenging times. I also want to express my deepest gratitude to my grandparents, Xuenong Yuan and Diqing Wang, for their persistent care since I was born. Without them, I would not even have had the chance to attend a relatively good junior high school. Last but not least, I would like to thank my younger brother, Kaiyue Wang, for being a sensible and intelligent kid and keeping my parents company in my absence.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 VLSI Physical Design

Since the introduction of integrated circuits (ICs) in 1958 [1], there has been a remarkable surge in the number of transistors integrated onto these chips. What started with no more than tens (10^2) of transistors has escalated to billions, as exemplified by the Apple M2 Ultra processor featuring a staggering 1.34×10^{11} transistors [8]. This exponential growth in integration has revolutionized the landscape of computing devices, enabling the creation of smaller yet highly efficient devices that have become ubiquitous in our daily lives, such as laptops, digital cameras, and mobile phones. Alongside the benefits of higher integration levels, more challenges are brought to the physical design of very large-scale integrated (VLSI) circuits.

Physical design is a critical part of the modern electronic design automation (EDA) flow, as depicted in Figure 1.1. After logic synthesis converts the functional description of a circuit into an optimized gate-level netlist, physical design will work out a physical layout based on the abstract logical description. Placement and routing (P&R) are two major components of physical design. Place-

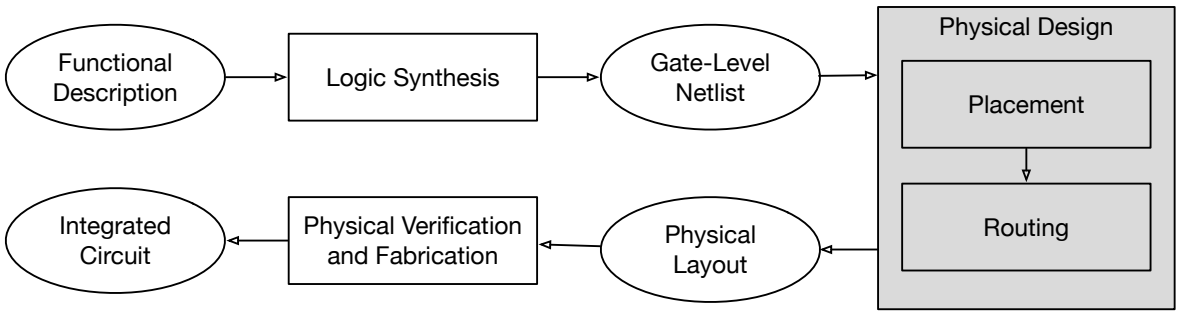


Figure 1.1: *A simplified VLSI circuit design flow. The detailed one is covered in [59].*

ment decides the location of each circuit component, with the goal of enhancing timing and routability in subsequent stages. On the other hand, routing generates physical connections between pins, adhering to design rules mandated by the manufacturing process.

Routing is a highly complicated process where (i) multiple objectives like wire length, via number, delay, and power need to be optimized, (ii) design rules need to be followed, (iii) millions or billions of nets need to be connected and they compete for the limited routing resource. Featuring the above, routing has been one of the most time-consuming process in the VLSI physical design flow. To reduce the complexity and shorten the runtime, routing is further decomposed into two steps, namely global routing and detailed routing. By global routing, a coarse-grained routing plan will be generated, minimizing the total wire length, the number of overflows, or other objectives. After that, guided by the result of global routing, detailed routing generates the actual wires and vias while taking care of the pin shapes and honoring specific design rules (e.g., min-area rule, various spacing rules, etc.). With a good global routing solution that does not have congestion issue, detailed routing can be finished much faster and with better solution quality.

1.2 Challenges

Compared with several decades ago, physical design is now facing problem instances that are orders of magnitude bigger. It is also constrained by a larger number of design rules associated with deep sub-micron technology nodes. Both factors are making routability awareness more desirable in the process. In addition to traditional objectives like power, performance and area (PPA), more attention is drawn to potential security threats posed by fabless manufacturing. In this thesis, we address the emerging routability and security challenges in VLSI physical design.

1.2.1 Misalignment Between Placement and Routing Objectives

Ideally, to achieve the best PPA of circuits, placement and routing should be conducted simultaneously. Facing the complex physical implementation problem, however, placement and routing are typically separated with different objectives to reduce complexity and to accelerate the design cycle. For instance, placement usually focuses on optimizing the half-perimeter wire length (HPWL) and estimations of routability, such as area density and pin density. On the other hand, global routing aims to minimize the routed wire length and number of overflows. As modern VLSI designs can involve over ten routing layers, each with different routing resources and various design rules, the correlation between the two-dimensional (2D) HPWL and the three-dimensional (3D) routed wire length is much weaker. Additionally, satisfying the estimated routability does not guarantee a routable design. In contrast, reserving a larger margin for routability can lead to longer wire length and delay. One of the recent trends is to conduct global routing during placement so that a congestion map can be generated to guide the placer [66, 29, 24], which provides

a more accurate model of routability. Placement and routing, however, are still optimized towards different directions in such frameworks. The misalignment between the objectives will inevitably lead to degradations in the final solution quality [51, 52]. Moreover, as the number of transistors rapidly grows, the adverse effect of separating the two processes can be further amplified.

1.2.2 Complex Design Rules and Pin Shapes in Detailed Routing

In detailed routing, metal wires and vias with geometric shapes are formed to connect pins, with the goal of minimizing the number of design rule checking (DRC) violations that can occur when objects in the physical layout are either too small (violating the min-area rule) or too close to each other (violating spacing rules). Since concurrent detailed routers typically suffer from scalability issues, modern state-of-the art detailed routers usually rely on heuristics like rip-up and reroute (RRR), history cost to iteratively resolve DRC violations. With more advanced technology nodes, routing around pins can become intricate due to complicated design rules and off-grid pin shapes [96, 60]. Routing resources near pins can be highly limited, and the RRR approaches cannot guarantee that conflicts between pins will be eventually resolved. Besides, the actual spacing requirements from certain design rules cannot be determined until the routing solution is constructed (e.g., parallel-run length spacing). As routed wires or via metals can form polygons with new outlines, some spacing violations can be difficult to detect, leading to additional rounds of RRR and prolonged runtime.

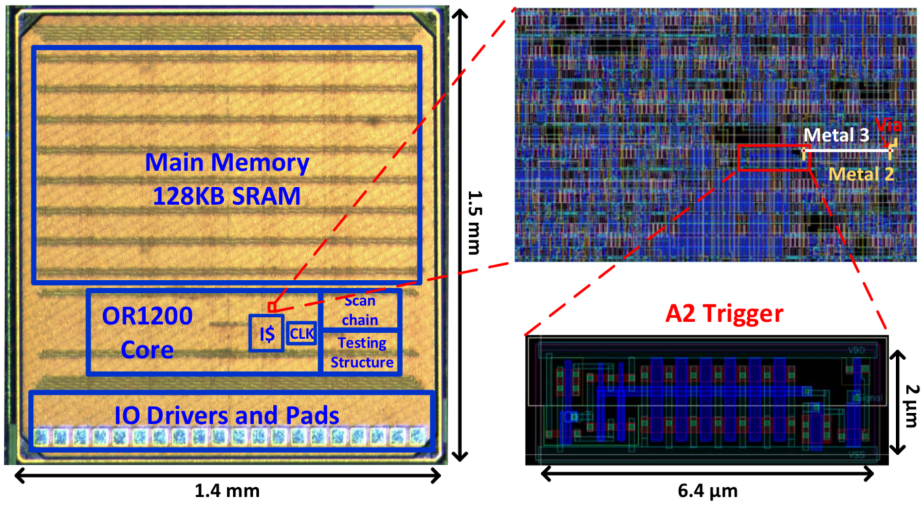


Figure 1.2: *An A2 Trojan that can be inserted into the physical layout during fabrication time [128].*

1.2.3 Hardware Trojans in Physical Layouts

Due to cost benefits, the supply chains of ICs are largely outsourced nowadays. Passing ICs through various third-party providers, however, gives rise to many threats, such as piracy of IC intellectual property or the insertion of hardware Trojans, which has been found in commercial chips [112]. By design, hardware Trojans are minor in extent, but severe in fallout [121, 32, 128]. For instance, a Trojan can leak information from an IC, reduce its performance, disrupt its functioning, etc. Figure 1.2 shows an analog Trojan that is extremely small and can be inserted into the physical layout during fabrication. When activated, it will change the victim flip-flop to any desired value and perform a privilege escalation attack [128]. The existing industrial physical design tools, however, primarily focus on PPA and are not designed to handle such threats. With unused placement and routing resources, various Trojans can be inserted stealthily. Although numerous countermeasures against hardware Trojans have been proposed over the years, most of them fall short in terms of resilience against advanced attacks and/or overheads.

1.3 Thesis Overview

Given the aforementioned challenges, this thesis presents a set of methodologies for improved routability and security in physical design.

In Chapter 2, we review related works on placement and routing co-optimization, detailed routing, and security closure in physical layouts.

In Chapter 3, to bridge the gap between placement and routing (P&R), we propose an efficient P&R co-optimization framework that conducts cell movements and reconnects broken nets by A* search-based partial rerouting. A lookup table-based approach is proposed for accurate wire length gain estimation, which further guides the partial rerouting process. We show that the framework is highly efficient and can effectively reduce the total wire length of an initial layout without introducing congestion issues.

In Chapter 4, we introduce a fast and robust pin access analysis framework. A novel sweep line-style algorithm is adopted for quick design rule checking and inter-route conflict detection. We also propose an SAT formulation for the conflict-free route selection problem and utilize an incremental SAT solving technique to obtain a pin access scheme that is not only DRC-clean but also optimized towards custom objectives. We further integrate FastPass into the state-of-the-art academic detailed router, Dr. CU. With integration of FastPass, Dr. CU is able to produce detailed routing results with much less short area and fewer DRC violations.

In Chapter 5, we present methods to harden the physical layouts of ICs against post-design insertion of hardware Trojans. We propose a multiplexer (MUX)-based locking scheme devised to withstand the state-of-the-art machine learning (ML)-based attacks on locking, and a physical design flow that is capable of hardening any post-route layout with controllable timing and area overheads.

Chapter 2

Literature Review

In this chapter, we will review the literature on several topics related to physical design, including placement and routing co-optimization, detailed routing and security closure against hardware Trojans.

2.1 Placement and Routing Co-optimization

To solve the complex physical design problem, placement and routing (P&R) are typically solved as two separate problems, which can be further decomposed into global placement, legalization, detailed placement, global routing and detailed routing [59].

In modern **global placement**, the layout region is usually divided into regular placement bins and a quadratic [35, 17, 118, 115, 45, 67, 66, 76, 46, 29] or non-linear [95, 64, 16, 23, 85, 86, 53, 24, 77, 80] optimization problem will be formulated to optimize the total HPWL. Meanwhile, area density constraint, which specifies that the total area of the cells in each bin should be optimized towards a target density value to spread out the cells, is commonly used to reserve some margin for routing

closure. Pin density is often considered to capture local connections that do not span across bins. After global placement, cells are roughly distributed in the layout region with some overlaps. **Legalization** will take place to remove overlaps and to find each cell a legal location in the placement rows while introducing minimum disturbance to the global placement result [114]. Based on a legalized placement result, **detailed placement** continues to optimize objectives like HPWL, routability by techniques like cell shifting, swapping and re-ordering [102, 101, 28, 130, 83, 31].

In **global routing**, the layout is usually sliced by evenly distributed horizontal and vertical lines to form a 3D grid graph, where each grid is called a GCell. Each GCell or edge between abutting GCells[1] will be associated with a supply and a demand value. Overflow can happen when the demand exceeds supply, indicating possible routability issues in the later detailed routing stage. Global routers mainly focus on generating a coarse-grained routing solution where the total routed wire length and overflows are minimized[2]. Existing global routing approaches can be roughly divided into two categories according to the routing flow: (i) 2D global routing with layer assignment and (ii) 3D global routing. In the first category, multiple routing layers will be compressed into a single-layer routing grid. Based on the grid, an optimized 2D routing solution will be generated by a 2D global router and then projected to 3D space by layer assignment. Due to its efficiency, this flow has been adopted in many global routers such as NTHU-Route 2.0 [20], MAIZEROUTER [89], FGR [104], NTUgr [22], FastRoute 4.0 [127], NCTU-GR [27, 82] and TritonRoute-WXL [61]. In the second category, global routing is directly conducted on the 3D grid graph [104, 120, 126, 79]. For example, GRIP [120] uses integer programming to choose routing topology for all nets simultaneously

[1]In global routing, some resource models are GCell-based while some others are edge-based.

[2]Timing and power are also part of the objectives sometimes.

such that wire length and via cost can be minimized subject to the edge capacity constraints. Despite the good solution quality, the concurrent approach suffers in scalability. CUGR [79], which is one of the state-of-the-art academic sequential routers, first decomposes multi-pin nets into two-pin nets by FLUTE [25] and then conducts 3D pattern routing which performs pattern routing and layer assignment simultaneously. With dynamic programming, an optimal 3D routing solution can be produced for each net. After that, multi-level 3D maze routing-based rip-up and reroute is conducted to reduce overflows. Recently, GPU-acceleration also gains more popularity [74, 81, 75]. Guided by the global routing result, **detailed routing** is conducted to generate the actual wires and vias while considering more detailed design rules and exact geometric shapes. In Section 2.2, we will give a more comprehensive review about detailed routing.

Such a divide-and-conquer paradigm keeps the physical design problem manageable. However, the misalignment in objectives and constraints can lead to slow design closure and sub-optimal solution quality [51, 52]. For example, HPWL, which has poor proximity to the routed wire length in modern designs [103], is commonly adopted as the optimization objective in placement due to its computational efficiency [23, 115, 67, 76, 85, 29, 46, 53, 24, 77, 80]. ROOSTER [103] proposes to optimize the rectilinear Steiner minimum tree (RSMT) length during global placement since it provides a more accurate estimation than HPWL. However, as modern VLSI routing is conducted in a 3D space, the correlation between the 2D estimations (HPWL, RSMT) and the 3D routed wire length can be much weaker with the growing number of heterogeneous routing layers. For instance, by the 32 nm technology node, there can be twelve metal layers with at least four different metal widths and even more variation in thickness [5]. Additionally, reserving a margin for routability by rough estimations like area density and pin density in the

placement stage can lead to long routed wire length and delay. Many efforts have been made to narrow the gap between placement and routing.

2.1.1 Placement with Routing

Prior to the ISPD 2007 and ISPD 2008 Global Routing Contest [93, 92], many publications held the belief that incorporating global routing into a placement flow was excessively time-consuming, despite its potential for providing more accurate routability estimation that reflects actual routing behavior. However, numerous efficient global routers [100, 99, 20, 89, 127, 22, 27, 126, 82] have been developed since then, enabling the integration of placement and routing.

Several placement frameworks employ global routing to obtain more accurate congestion information. CRISP [105], Eh?Place [29] and RePLAce [24] utilize the global routing congestion map to guide cell inflation such that cells in congested regions can spread out effectively during global placement. Similarly, SimPLR [66], Eh?Placer [29] and NTUplace4dr [53] dynamically adjust the target density in placement bins based on the congestion map to improve routability. In CROP [130], a placement refinement framework is proposed, which interleaves congestion-driven module shifting and congestion-driven detailed placement. Both techniques are guided by congestion information from a simple global router that considers L/Z pattern routing and 3-bend routing.

In addition to congestion information, other approaches also leverage the routing topology for placement optimization. IPR [101] explicitly maintains a global routing solution generated by FastRoute [100]. At the end of the global placement step, routability driven refinement (RDR) is proposed to improve congestion by iterative cell movement and rerouting. During RDR, IPR determines the next step movement for a cell by moving it to each of the eight nearest bins, rerouting the nets, so that

the optimal location that balances congestion reduction and bin utilization can be found. Yet, as each round of such movement evaluation requires rerouting all the nets associated with the moved cell, IPR is rather time-consuming and is hard to deploy in practice.

In GRPlacer [28], global routing is conducted to measure congestion and split multi-pin nets into two-pin connections, which are weighted based on the average congestion within their bounding boxes. GRPlacer then considers the congestion-driven optimal region for cell shifting to reduce the routed wire length, which is approximated by the two-pin nets. Based on the two-pin net decomposition, GRPlacer also formulates a bipartite matching problem to facilitate multi-cell movement. However, estimations based on the two-pin net decomposition may lead to less accurate wire length optimization. For example, when a cell moves relatively far from its original location, a better net decomposition scheme may exist but the decomposition will not be updated until the global router is invoked again.

Instead of building a congestion map, Ropt [83] performs 2D routing based on a local-routability-aware cost function to find each cell a new destination with the lowest cost. In Ripple 2.0 [46], a simplified version of FastRoute is invoked to produce a routing solution and a congestion map. Unlike other approaches that only inflate cells in congested regions, Ripple 2.0 also inflates cells associated with nets routed across blockages. CR&P [2] utilizes a 3D global router to estimate the future routing cost of possible cell movements and then employs integer linear programming (ILP) to find the best destination for multiple cells. After cell movement, CUGR [79] will be invoked to reroute the nets.

The placement with routing paradigm is widely adopted in modern design practices. However, it does not mark the end of P&R co-optimization. While placers often leverage congestion information extracted from global routing solutions, the

exact routing topologies that capture actual routing behaviors are not fully utilized. Furthermore, in this paradigm, 2D routing is typically conducted for time efficiency while the actual routing happens in 3D space and can be constrained by various routing rules. Those factors contribute to a noticeable gap that still exists between placement and routing.

2.1.2 Routing with Cell Movement

Routing with cell movement, where placement is incorporated in global routing by allowing cell movement during the routing stage, is gaining more attention with the recent ICCAD 2020 and ICCAD 2021 Routing with Cell Movement Contest [51, 52] and is expected to further bridge the gap between placement and routing.

(a) Sequential Methods: With a placed and routed layout, SRP [44] identifies problematic cells that have nets routed across congested areas, relocates those cells, and rebuilds connections. To find a new destination for each problematic cell, SRP first deletes part of the nets associated with the cell to form a set of subnets. Starting from the remaining segments, a multi-source propagation step is conducted for each net of the cell, and GCells where all the nets meet first are treated as possible cell movement candidate locations. For each possible location, the cell is relocated, and the broken nets are reconnected using a multi-subnet maze routing algorithm. However, as pointed out in [134], the target location derived from the proposed multi-source propagation algorithm does not necessarily minimize the total routed wire length, even when the routing resource is abundant.

Recently, a few works have attempted to narrow the gap between placement and routing based on the more formal problem formulation in the two ICCAD contests [51, 52]. In [134] and its preliminary version [135], several techniques

are proposed within a framework which iteratively performs cell movement gain estimation, cell movement and rerouting. To find potentially good destinations for cell movement, a multi-net-based location prediction step is introduced. When estimating the wire length reduction by moving a single cell to different locations, routing segments connected to the cell are ripped up, and the remaining routing topology is used to calculate the cost of reconnecting the moved cell. Locations with the highest reduction in routed wire length are considered for future cell movement. To accelerate the reconnection cost estimation, a look-up table (LUT) will be preprocessed to enable constant-time calculation of the minimum routing distance between two 3D points under various routing constraints, such as minimum routing layer and preferred routing direction. Besides, by limiting the search space and optimizing the data structures, a fast congestion-aware 3D maze routing algorithm is developed for this routing-intensive scenario. Huang et al. [54] propose a breadth-first search (BFS)-based algorithm to identify candidate locations for cell movement. This algorithm mimics the routing process instead of solely focusing on wire length optimization. It is expected to better correlate with the rip-up and reroute process after cells are moved, albeit at the cost of longer runtime.

Based on the ICCAD 2020 contest benchmark suite [51], the ICCAD 2021 contest [52] further incorporates power and timing to formulate a weighted routed wire length objective, where each routing layer is assigned a power factor and each net has a timing criticality factor. Besides, a simpler cell blockage demand and a voltage area constraint that limits the cell movement locations are introduced. In [129], a two-level framework called ATLAS is proposed to address this advanced version of the routing with cell movement problem. At the first stage, it conducts a coarse-level cluster-based cell movement, where cells are clustered to minimize necessary via usage resulted from the min-routing-layer constraint. Cells in the

same cluster will then be moved as a whole. In the second step, fine-grained single cell movements will be conducted to refine the placement and routing solution. During the cluster-based and single cell movement process, a LUT-based approach is proposed to accurately measure the weighted wire length reduction of possible cell movement under various routing constraints. Based on this, an A* search-based partial rerouting algorithm is adopted for quick routing topology reconstruction. Besides, a layer weight adjustment scheme is proposed to encourage routing on higher metal layers, which is not only helpful for weighted wire length optimization but also relieves routing congestion on lower metal layers. Zhu et al. [132] propose another iterative flow where single cell movement and cluster-based movement are interleaved. An LUT similar to that in ATLAS [129] is used for efficient routing estimation. In their work, each pin will be mapped to the min-routing-layer of its associated net to form a virtual pin so that the constraint does not need to be considered during the routing process. Thanks to this, the LUT can be constructed with one less dimension compared with ATLAS [129].

(b) Concurrent Methods: Inspired by the routing with cell movement contests, ILP-GRC [37], an ILP model is proposed to move the cells and reroute the affected nets simultaneously. Specifically, a set of possible routing solutions will be generated for each net and for each possible cell movement location. ILP is then formulated to pick cell movement destinations and the corresponding routing solutions at the same time such that the total routed cost can be minimized without causing GCell overflows. To trade off runtime with quality, a dynamic paneling technique is adopted, where nets are divided into panels for parallel ILP solving.

2.2 Detailed Routing

Guided by a global routing solution in which wire length, congestion and some other objectives (e.g., timing, power, etc.) are optimized, detailed routing generates actual wires and vias, considering more detailed design rules and exact geometric shapes. Inspired by the ISPD 2018 and 2019 Initial Detailed Routing Contest [87, 84], many academic detailed routers have been developed recently, such as the TritonRoute series [62, 63, 61], Dr. CU [21, 73], SmartDR [41] and the one mentioned in [116], along with new algorithms for high-quality and efficient detailed routing. A competent detailed router typically incorporates a subset, if not all, of algorithmic components for (i) pin access analysis, (ii) track assignment, (iii) design rule checking (DRC) and (iv) design rule-aware path search. In this subsection, we will review each of the components, highlighting the recent developments in pin access analysis and design rule-aware path search algorithms.

2.2.1 Pin Access Analysis

With the scaling of technology nodes and higher level of integration, standard cell pin access has become non-trivial due to complicated design rules and off-grid pin shapes [96]. Under such extreme circumstances, as an early step in a detailed routing flow, pin access analysis will be responsible for planning violation-free paths from pin shapes to some easily accessible grid points. This ensures that in the later path searching stage, routing can land on on-track locations without causing possible conflicts between pin access segments (i.e., metal wires and vias) and fixed metals (e.g., blockages, pin shapes, etc.).

In recent years, there has been a proliferation of various approaches for pin

access, which can fall into two categories according to the extent of pin access planning: (i) intra-cell pin access and (ii) inter-cell pin access.

(a) Intra-Cell Pin Access: Nieberg [96] first introduces the idea of circuit class to reduce redundant computation. Standard cell instances that (i) are of the same cell type, (ii) have the same offset with respect to the routing grid and (iii) have geometrically equal situations[3] are classified into one circuit class. For each circuit class, a two-step approach is conducted, involving the generation of design rule-compliant paths, followed by a conflict-free path search using branch-and-bound techniques. Possible inter-cell conflicts are left to the rip-up and reroute process in the sequential detailed routing stage. Xu et al. [123] address intra-cell pin access issues under self-aligned double patterning (SADP)-specific constraints and propose a standard cell layout optimization algorithm to maximize pin access flexibility. In [116], pin access points are selected sequentially, one pin after another. For a pin, a set of valid hit points[4] will be identified and then filtered by solving a maximum independent set problem, resulting in sparser distribution of the remaining points. However, if a pin can only be accessed by off-track vias, this approach will require other workarounds. Dr. CU [21, 73] utilizes L-shape routes and off-track vias to access pins that are hard to reach. Yet, without a comprehensive pin access analysis framework (PAAF), Dr. CU fails to capture DRC violations when complicated pin shapes are involved. SmartDR [41] introduces a scheme for flexible pin access during path search, where pin access paths are precomputed for each pin but will only be selected and instantiated during the path search stage.

[3]If those cells including the pre-routed wires can have the same layout by flipping and/or rotation, they are considered to have geometrically equal situations.

[4]Valid hit points refer to on-grid locations on which a via can be placed without causing DRC violation.

Intra-cell pin access approaches generally provide the path search step with more flexible pin access point choices. However, with more advanced technology nodes, routing around pins can be non-trivial due to the more complicated design rules, irregular pin shapes and highly limited routing resources. In such scenario, more choices can indicate a larger search space which is harder to explore. Intra-cell pin access, which does not consider pin access interference between neighboring cells or even between pins from the same cell, can therefore lead to unsolvable DRC violations and excessively long runtime.

(b) Inter-Cell Pin Access: Inter-cell pin access approaches further take the interaction between neighboring standard cells into account. Ozdal [98] introduces a multicommodity flow model aimed at identifying escape routes for pins in dense clusters. Xu et al. [125, 124] address pin accessibility under SADP constraints. In [125], both intra-cell and inter-cell pin access are precomputed, and standard cell pin access planning is incorporated into regular routing to enhance routability. Yet, the access points are chosen dynamically during the sequential routing stage, which sometimes fails to resolve resource competition, as pointed out in [124]. To address the pin access interference problem, Xu et al. [124] formulate concurrent pin access optimization as a weighted interval assignment problem and solve it using Lagrangian relaxation. Jiang [57] introduces a pin access-oriented concurrent detailed routing step with an integer linear programming formulation to optimize both pin access and local connections after track assignment.

Recently, Kahng et al. [60] present a design rule-aware PAAF that utilizes dynamic programming (DP) for both intra-cell and inter-cell pin access planning. The framework is shown to produce DRC-clean pin access schemes and greatly improves the detailed routing quality. However, the DP-based algorithm assumes that conflicts

only occur between adjacent pins in terms of their x-coordinates, which is not always true. As a result, each generated pin access pattern needs to undergo thorough validation by a DRC engine, which can slow down the process significantly.

Impact of Pin Access in Other Scenarios: Besides being a vital part of the detailed routing process that enables DRC-clean routing solutions, a good pin access methodology can play important roles in other physical design scenarios too, including but not limited to global routing, detailed placement, and even P&R co-optimization [31, 61, 58]. TritonRoute-WXL [61] incorporates a pin access engine [60] to enhance global routing by offering a more accurate estimation of the routing resources utilized for pin access. Ding et al. [31] present an approach for enhancing pin accessibility through detailed placement refinement. Their method utilizes a cost function to represent pin accessibility, without taking specific design rules into account. For more accurate pin accessibility evaluation, Kahng et al. [58] present an in-route placement refinement flow which conducts precise pin access analysis and significantly expedites DRC convergence.

2.2.2 Track Assignment

Track assignment was first investigated in the context of constrained via minimization [70, 108, 18]. Batterywala et al. [11] later introduced track assignment as an intermediate step to narrow the gap between global routing and detailed routing. In track assignment, wire segments, also called interval of global route (iroute), are extracted from global route guides, and each of them will be assigned to a specific routing tracks such that some cost function can be minimized. In this way, the majority of the wires can be implemented in the layout efficiently, which can either serve as a fast detailed-routability estimation or a relatively good starting point of

detailed routing.

In [11], the track assignment problem is formulated as a weighted bipartite matching problem and solved by an iterative heuristic. However, the approach can suffer in scalability and does not take local nets into account. Wong et al. [119] propose a negotiation-based track assignment algorithm, where iroutes will be extracted for both global and local nets. A rip-up and re-assign heuristic is employed to efficiently reduce a cost that accounts for wire overlaps and wire length. TraPL [109] further addresses the congestion issue in each GCell by (i) connecting local nets with single-trunk trees, and (2) estimating the via locations and partial track utilizations for better congestion estimation in track assignment. Despite achieving better solution quality than the negotiation-based approach [119], TraPL is much slower. Liu et al. [78] propose a negotiation-based track assignment algorithm which is aware of local nets, via locations and pin access, outperforming TraPL in both solution quality and runtime. Zhuang et al. [133] point out that few existing approaches consider design rules in track assignment and address this issue with a design-rule-driven track assignment algorithm.

2.2.3 Design Rule Checking

To verify the correctness of a detailed routing solution, design rule checking (DRC) is necessary. After nets are routed, design rule checking will take design rules (e.g., min-area rules, spacing rules, etc.) and geometric shapes (e.g., wires, via enclosures, cuts, pin shapes, blockages, etc.) in the layout into account, and reports the location, rule type and related objects of the detected violations. Computational geometry algorithms [12, 71, 107, 13, 43] are commonly used for efficient design rule checking.

Only a limited number of works have presented comprehensive studies on DRC. Kahng et al. [61] disclose the implementation details of a geometry-based DRC

engine, which is integrated into an open-source router[5]. He et al. [47] introduce a GPU-accelerated DRC algorithm which achieves an impressive speedup over CPU-based DRC algorithms.

2.2.4 Design Rule-Aware Path Search

During track assignment, many design rules are either ignored or considered with simplifications for time efficiency. Thus, a number of DRC violations can exist after track assignment and need to be resolved by a more fine-grained routing process, where path search algorithms that are aware of exact design rules can be indispensible to construct DRC-clean routing solutions.

(a) Sequential Methods: Under this category, net-by-net routing is conducted, and the previously routed nets will be treated as obstacles. Based on that, rip-up and reroute is typically adopted to resolve DRC violations progressively.

In the single net routing context, extensive research has been done on path search algorithms that aim at efficiently finding a minimum cost path from a source to a single or multiple sinks based on a grid graph or a general graph [30, 72, 97, 113, 48]. Gonçalves et al. [39] deliver a more detailed survey on these conventional algorithms, which serve as the foundation of many modern design rule-aware path search algorithms [19, 3, 21, 40, 63].

MANA [19] is capable of finding shortest paths under the end-end separation rule and the minimum wire length rule. BonnRoute [3] incorporates an interval-based shortest path algorithm, which utilizes a multilabel system to avoid same-net violations. Yet, the multilabel path search is also reported to be much slower than standard path search. Dr. CU [21, 73] transforms the min-area rule into a

[5]The source code is available at https://github.com/ABKGroup/TritonRoute-WXL.

minimum-length constraint and extends the Dijkstra's algorithm [30] to capture min-area violations and to consider possible fixes during path search. Different from MANA [19], this algorithm does not enforce minimum length as a hard constraint and thus gives a smoother rip-up and reroute process. Gonçalves et al.[40] introduce an interval-based A* search algorithm, which considers via types and various design rules such as min-area and same-net cut spacing. In TritonRoute [63], an A* search-based algorithm is presented, which utilizes a preprocessed look-up table for efficient same-net design rule checking.

Due to the greedy nature of the sequential methods, history cost is commonly adopted to penalize routing in the neighborhood of past DRC violation regions to avoid creating the same problematic route repeatedly. Sequential methods have now been widely adopted both in academia and industry because of its good scalability.

(b) Concurrent Methods: While sequential methods typically route one net after another, concurrent path search algorithms constructs routing solutions for multiple nets simultaneously.

RegularRoute [131] conducts routing in a bottom-up layer-by-layer manner. For each layer, the layout region is divided into panels, where the global segment assignment problem is formulated as a maximum weighted independent set problem and solved by a quick heuristic. In a similar bottom-up framework, Kahng et al. [62] propose an ILP-based algorithm for routing in each panel. In MCFRoute [55, 56], a multi-commodity flow-based ILP model is introduced for detailed routing. The proposed formulation not only takes various design rules into account but also maximizes the usage of redundant vias, which is preferred for higher yield.

2.3 Security Closure Against Hardware Trojans

Trojans are malicious hardware modifications [121, 32] with a wide range of purposes including: (i) leaking information from an IC, (ii) reducing its performance or (iii) disrupting its functionality. Such modifications can be introduced through untrustworthy third-party intellectual property (IP) or adversarial designers during outsourced mask generation, manufacturing or packaging of ICs and can be triggered either internally or externally. Most, if not all, Trojans consist of a trigger and a payload. The trigger activates the payload upon meeting certain attack conditions, and the payload executes the actual attack. Triggers often rely on low-controllability nets (LCNs) to complicate their detection during testing, while payloads target sensitive assets such as key registers. Note that trigger and payload are implemented individually but work together in tandem. Additionally, most Trojans require layout-level resources such as open placement sites, free routing tracks, and/or available timing slacks, which suggests potential direction for Trojan defense.

Security closure, as an emerging paradigm, aims to proactively harden the physical layouts of ICs at design stage against various threats that may arise post-design [69, 68, 36]. In the context of Trojan prevention, security closure involves controlling physical synthesis to make Trojan insertion impractical while managing the impact on design quality (i.e., PPA) [69, 68]. For example, an aggressively dense layout may leave few placement and routing resources exploitable for Trojan insertion. While aggressively dense layouts are already challenging from a design quality standpoint, such naive approach may not suffice for security closure. This is because advanced Trojans like A2 [128] may require only 20 placement sites[6] for an

[6]This is a remarkable exception—other Trojans reported in the literature, as well as a digital version of A2 itself, require hundreds or thousands of sites [117].

advanced analog implementation [117], and such a small number of sites may still be available even in aggressively dense layouts. Furthermore, adversaries may attempt second-order attacks by revising the layout to create space for Trojan insertion. Hence, effective Trojan prevention approaches are critical for security closure in layouts. While various types of Trojan defense methods have been proposed, this thesis focuses on locking-based Trojan prevention and layout-level Trojan prevention.

2.3.1 Locking-Based Trojan Prevention

Logic locking, also known as locking, involves the incorporation of key-gate structures controlled by secret key-bits. While locking is commonly used for protecting the intellectual property (IP) of ICs, it can also be utilized for Trojan defense [33, 106, 88, 111]. Dupuis et al. [33] lock low-controllability nets (LCNs) using AND/OR key-gates to hinder insertion of Trojan triggers. They consider timing slacks, varying toggling thresholds, and balanced switching probabilities. However, their approach may fail against the insertion of targeted Trojans[7] because they focus on tuning the controllability of nets with AND/OR key-gates rather than obfuscating the design.

Samimi et al. [106] follow similar principles as Dupuis et al., but utilize X(N)OR key-gates. Marcelli et al. [88] propose a multi-objective algorithm, seeking to minimize LCNs and to maximize the efficacy of X(N)OR locking simultaneously. Šišejković et al. [111] secure inter-module control signals against software-controlled hardware Trojans using encryption circuitry along with regular locking. However, their approach does not protect intra-module signals.

The above prior art has the following limitations in common. First, all of them consider Trojan prevention at the netlist level instead of layout level, which

[7]Targeted Trojans are Trojans that require the understanding of the original design.

means they cannot conclusively prevent the insertion of Trojan logic into the layout. Besides, none of them explicitly prevent Trojan payloads. While Šišejković et al. [111] use locking in general without focusing on triggers or payloads, the others only lock LCNs to hinder the insertion of Trojan triggers. Additionally, none of them demonstrate robustness against machine learning (ML)-based attacks, which will be discussed shortly. Therefore, these locking defenses are susceptible to second-order attacks that can bypass the locking mechanisms. For example, Dupuis et al. [33] employs a direct, hard-coded correlation of key-bit "0" to OR key-gates and key-bit "1" to AND key-gates, respectively. This correlation can be exploited by adversaries to circumvent the defense.

Recently, ML-based attacks like SAIL [15], SCOPE [4], OMLA [7] and MuxLink [6] have succeeded in learning various design features and predicting key-gates, all in an oracle-less setting[8]. It means that they can achieve these results without requiring a functional chip that can be queried for its functional behavior. ML-resilient locking schemes remain an open challenge but are essential for locking-based and proactive Trojan prevention.

2.3.2 Layout-Level Trojan Prevention

Xiao et al. [122] propose filling layouts with built-in self-test components, arguing that adversaries' attempts to tamper with these structures and regain layout resources for Trojan insertion can be detected through post-silicon testing. However, the detection process requires 100% test coverage for full efficacy, which can be difficult to achieve for large and complex designs. The methodology is also challenged by high utilization rates, limiting its practicality. Similarly, Ba et al. [10, 9] suggest filling

[8]In this thesis, our focus is on proactive, pre-silicon prevention of Trojans, not for IP protection against end-users. Therefore, we solely consider the oracle-less setting.

layouts with additional circuitry, considering priorities for empty spaces, routability, etc. In [122, 10, 9], the number of additionally required primary inputs scales with layout filling, which is impractical. Pads for primary inputs/outputs (PIs/POs) in actual ICs are large, and if not employed wisely, they can significantly increase the chip outline, directly increasing the cost for silicon area.

Knechtel et al. [69] propose techniques for Trojan-resistant physical synthesis based on locally increasing placement density (to hinder Trojan insertion) and locally increasing routing density (to hinder Trojan routing). Hossein-Talaee et al. [49] redistribute white space and open sites that could be exploited for Trojan insertion. Stimulated by the recent ISPD 2022 contest [68], Hsu et al. [50] propose a greedy cell movement algorithm to eliminate exploitable regions[9]. Similarly, in the contest setting, Guo et al. [42] present a PPA-aware Trojan defense framework, which incorporates two algorithms to eliminate exploitable regions while minimizing the impact on circuit performance. First, a row-based cell shifting technique is employed to partition exploitable regions into smaller pieces while limiting the displacement of cells. A partitioning-based approach, involving buffer insertion and gate resizing, is then utilized to progressively decompose exploitable regions.

The above layout-level countermeasures have the following limitations in common. First, none of them protect against the insertion of targeted Trojans. In the absence of locking or other obfuscation schemes, the original design remains fully accessible to adversaries. Besides, none of the approaches conclusively demonstrate robustness against second-order attacks, where adversaries regain layout resources despite the defense measures. For example, local movement of standard cells in [50] can be easily reverted, and inserted buffers in [42] can be removed without affecting

[9]In the ISPD contest, exploitable region refers to a connected component of unused placements regions that contains more than 20 sites.

the circuit functionality.

Based on the above analysis, we identify a few desired properties for Trojan countermeasures. First, a defense should be robust against attempts from a foundry-based adversary to circumvent it before inserting their Trojan. Second, a defense should protect a design against various Trojans in general, for example, by limiting the layout resources that are exploitable for Trojan insertion (e.g., open placement sites, free routing tracks, available timing slacks) and disguising netlist structure and functionality, which is studied by adversaries for targeted Trojan insertion. Besides, any defense should incur limited, controllable overheads in design metrics, i.e., PPA.

Chapter 3

Efficient Placement and Routing Co-Optimization

In this chapter, we present Starfish[1], a placement and routing (P&R) co-optimization engine that minimizes the actual global routing wire length by cell movement and net rerouting. Meanwhile, all hard constraints under the routing with cell movement context are strictly followed (e.g., connectivity, min-routing-layer, overflow-free, etc). Our contributions can be summarized as follows.

- An efficient multi-threaded P&R co-optimization engine is proposed, which integrates several novel techniques to effectively minimize the total routed wire length based on an initial P&R solution.
- Our engine integrates a rectilinear Steiner minimum tree (RSMT)-guided maze routing technique for both wire length and runtime reduction.
- A lookup table-based approach is proposed for accurate wire length estima-

[1]Starfish, or sea stars, are well recognized for their marvelous ability to regrow arms. Inspired by the similarity between arm regeneration and the process of reconnecting a moved cell back to the existing routing segments, we name the proposed engine Starfish.

tion, which takes routing constraints such as min-routing-layer and preferred routing direction into account. With accurate point-to-point wire length estimation, a cell movement gain estimation scheme, which utilizes the existing net routing topologies, is proposed to pick good candidate destinations for cells. We further accelerate the estimation scheme by a median box-based pruning technique.

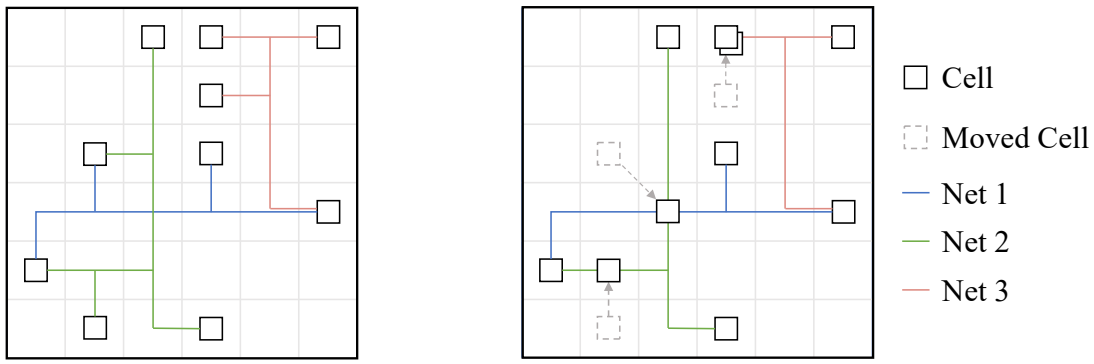
- In alignment with the proposed gain estimation scheme, we design a multi-source single-target A* search-based partial rerouting algorithm to quickly connect a moved cell back to the trunk of a previous routing topology.
- Experimental results on the ICCAD 2020 contest benchmarks suite [51] show that Starfish outperforms all the contestants. In particular, compared with the champion of the contest, Starfish can achieve 0.9% better scores[2] with 87% less runtime on average.

The rest of the chapter is organized as follows. In Section 3.1, we introduce basic ideas and the problem formulation of routing with cell movement. In Section 3.2, we present the core algorithms and techniques in Starfish. In Section 3.3, a series of experiments are conducted to demonstrate the efficiency and effectiveness of the proposed P&R co-optimization engine.

3.1 Preliminaries

In the routing with cell movement problem [51], a bounded number of cells can be relocated to improve the original placement and routing solution such that the routed

[2]In the contest setting, this can be considered as a significant gap, given that the differences in score between the first place and the other two top-3 participants are only 0.8% and 1.0% respectively on average.



(a) Before Routing with Cell Movement (b) After Routing with Cell Movement

Figure 3.1: *Illustration for routing with cell movement. Both (a) and (b) have the same HPWL, but the routed wire length in (b) is shorter.*

wire length can be further minimized without causing routing overflows. Figure 3.1 shows an example of routing with cell movement based on an existing placed-and-routed circuit. After cell relocation and net rerouting (shown in Figure 3.1(b)), a better P&R solution is achieved with shorter routed wire length. It is worth noting that, since this problem is not aimed at generating a totally different placement solution, the total number of cells that can be moved is limited (e.g. 30% of the total number of cells). In the following discussion, we denote C as the set of cells and N as the set of nets.

3.1.1 Coordinate Plane and Grid Graph

Coordinate plane and grid graph are two basic concepts in this problem, which represent the placement region and the global routing region respectively.

(a) 2D Coordinate Plane: Given the number of rows Row and the number of columns Col, the coordinate plane $P \subseteq \mathbb{Z}^2$ contains $|P| = Row \times Col$ coordinates and serves as the placement region. Each cell $c_i \in C$ locates at a specific 2D coordinates $p_i \in P$, and each 2D coordinates $p \in P$ can usually accommodate more

than one cell as long as the constraints described in Section 3.1.4 are satisfied.

(b) 3D Grid Graph: Similar to traditional 3D global routers [82, 126, 79], a 3D grid graph $G(V, E)$ is constructed to represent the 3D routing region. In this problem, V denotes the set of all global routing cells (GCells) in the 3D routing region. The size of $V \subseteq \mathbb{Z}^3$ is given by $|V| = Row \times Col \times L$, where L is the number of metal layers. Meanwhile, E is the set of edges showing the connectivity between GCells. There are two types of edges between GCells. One is the wire edge, which represents the intra-layer connection between two GCells lying on the same layer and being adjacent in the preferred routing direction of that layer. The other one is the via edge, which represents the inter-layer connection between two adjacent GCells locating at the same 2D coordinate.

3.1.2 Net Model

In the routing with cell movement context, each net possesses its own sets of routing segments to connect the pins, and the routed wire length of a net is computed as the total number of GCells traversed by its routing segments. In addition, the routing segments of each net must satisfy a min-routing-layer (MRL) constraint, which specifies a minimum 2D routing layer. The horizontal and vertical routing segments of a net $n \in N$ must occur on or above its min-routing-layer $mrl(n)$.

3.1.3 Overflow GCell Definition

To define an overflow GCell, we first introduce GCell capacity. The capacity of a GCell $v \in V$ is defined as the difference between its supply and demand, denoted as $cap(v) = supply(v) - demand(v)$. The term $supply(v)$ is a given value that measures the available resources for cell placement and net routing, and $demand(v)$

is the summation of cell blockage demand, cell extra demand and routing demand. The detailed formulation of $supply(v)$ and $demand(v)$ are given in [51], including how the cell extra demand is calculated. When the capacity of a GCell v is smaller than zero, that is, $cap(v) < 0$, v is considered as an *overflow* GCell.

3.1.4 Problem Formulation

Given a placed-and-routed circuit, the routing with cell movement problem is defined to minimize the total routed wire length by cell relocation and net rerouting. The final solution should satisfy the following constraints.

(1) *Max-Cell-Movement*: The total number of cells that can be moved is limited by a constant $MaxCellMove$.

(2) *Min-Routing-Layer*: Routing segments of each net must occur on or above its min-routing-layer.

(3) *Connectivity*: No open net exists.

(4) *Preferred Routing Direction*: The intra-layer routing segments must follow the preferred routing direction.

(5) *Overflow-Free*: No overflow GCell exists.

Satisfying the aforementioned constraints, the score of the final output solution is defined as:

$$Score = TWL_{input} - TWL_{output}, \tag{3.1}$$

where TWL_{input} and TWL_{output} denote the total wire length in the given input solution and the final output solution respectively.

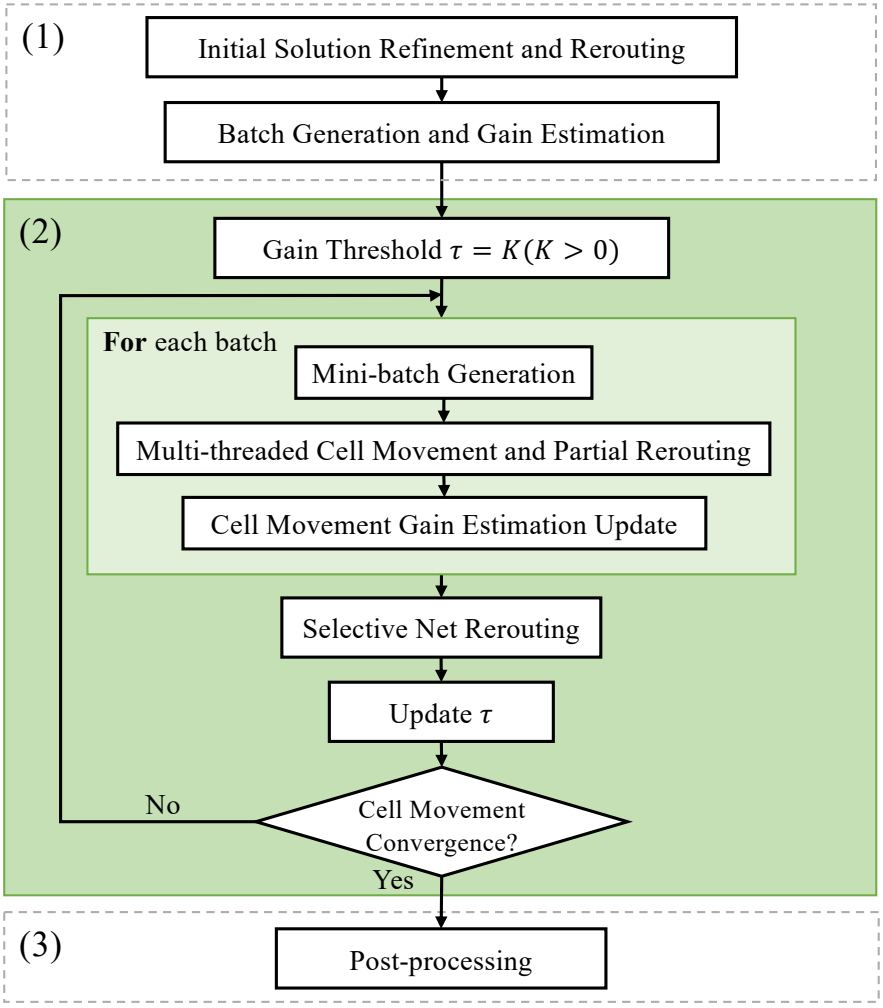


Figure 3.2: *Overall flow of Starfish. Our proposed co-optimization engine consists of three stages: (i) pre-processing, (ii) multi-threaded cell movement and (iii) post-processing.*

3.2 Methodology

3.2.1 Overview

As Figure 3.2 shows, our proposed algorithm can be divided into three major parts, (i) pre-processing, (ii) iterative multi-threaded cell movement, and (iii) post-processing.

In the pre-processing stage, we will first read in a given overflow-free routing

solution. As cycles and unnecessary wire segments may exist in this initial solution, a refinement step will first be conducted to generate a clean routing topology for each net.

After refining the initial solution, a congestion-driven rip-up and reroute (RRR) algorithm will be applied to greedily improve the routing solution, thus providing a better initial solution for the coming multi-threaded cell movement.

Before cell movements are performed, movable cells will first be partitioned into several batches, where cells in the same batch are independent from each other. Two cells are independent if they are not connected by any net. We then utilize a lookup table-based estimation scheme to calculate the cell movement gains and locate good candidate destinations for each movable cell.

In the iterative multi-threaded cell movement stage, cells will be moved and reconnected by an A* search-based partial rerouting algorithm that aligns with the proposed estimation scheme. In order to conduct cell movements in parallel, cells in an independent batch will be further partitioned into several mini-batches in such a way that cells in the same mini-batch do not overlap in terms of placement and routing region.

After handling an independent batch of cells, movement gains will be updated for cells that have been successfully moved or have connections with some moved cells. Note that this gain update step can be delayed until after an independent batch of cell movements is finished and can be readily parallelized, since the relocation of one cell will not affect the cell movement gains of other cells that are independent with it. A net rerouting scheme will then be applied to further optimize nets that have been partially rerouted during the previous cell movement step.

One may wonder about the cell ordering, which can have a significant impact on the solution quality. Indeed, with the two-level batching strategy, cells are

not explicitly ordered in our engine. Instead, we introduce a gain threshold τ to maintain a cell ordering implicitly. A cell movement will be committed only when its wire length gain is at least τ. By starting with a relatively large threshold and gradually decreasing its value, cells with higher potential of wire length reduction can be prioritized.

In the post-processing stage, steps are taken to further reduce the routed wire length while ensuring that the output solution is legal. As the cell ordering for optimal P&R co-optimization is non-trivial, to achieve better results, we allow more than *MaxCellMove* cells to be moved in the cell movement stage, and design a scheme to carefully move cells that bring fewer gains back to their original locations. Finally, several rounds of wire length-driven rerouting will be performed to explore potential gains based on the final layout.

In Section 3.2.2, we will discuss an RSMT-guided maze routing approach, which is adopted in all net rerouting stages except the partial rerouting after cell movement. In Section 3.2.3, details about the proposed cell movement and partial rerouting scheme will be covered. After that, we introduce the post-processing techniques in Section 3.2.4.

3.2.2 RSMT-guided Maze Routing

In the proposed flow, we utilize maze routing to improve the existing routing solution under a fixed cell placement. The routing function will be called many times throughout the entire co-optimization process. For instance in Figure 3.2, this RSMT-guided maze routing will be invoked in the initial rerouting of the pre-processing stage, the selective net rerouting of the multi-threaded cell movement stage, and the wire length-driven rerouting of the post-processing stage. Therefore, an efficient method with good quality is essential.

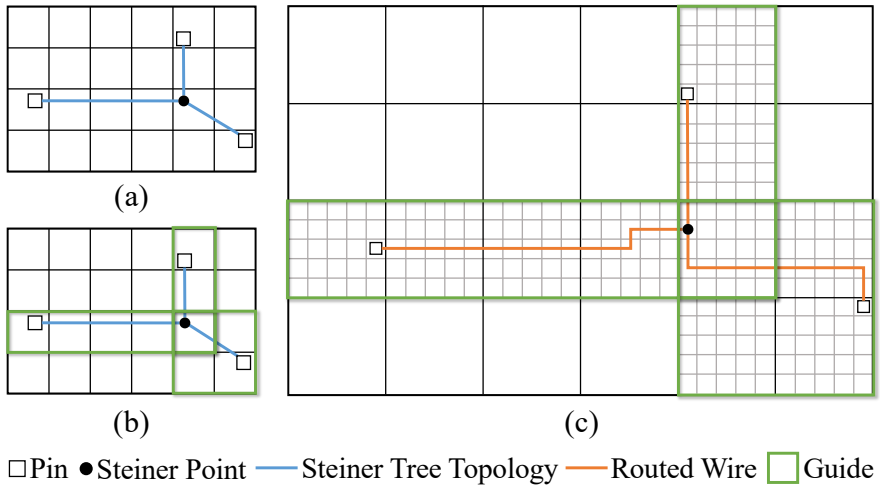


Figure 3.3: *Illustration for RSMT-guided maze routing. (a) An RSMT is generated according to the 2D layout by FLUTE [25]. (b) Route guides are generated according to the Steiner tree topology. (c) Fine-grained maze routing is conducted in the pruned solution space.*

Existing global routers like MGR [126] and CUGR [79] utilize multi-level routing to speed up the routing process, where a set of route guides will be generated to help pruning the routing solution space. With a similar idea in mind, we adopt an RSMT-guided maze routing approach, which is both faster and can achieve shorter wire length compared with a naive congestion-driven maze routing where routing is simply restricted in the net bounding box.

(a) Routing Region Generation: The guides will be generated based on a coarsen-grained grid graph. In our implementation, each block of 5×5 GCells forms a coarsened grid. As Figure 3.3(a) shows, an RSMT topology will first be generated by FLUTE [25] in an instant according the 2D coordinates of the pins. For an edge whose two end-points are not on the same row or column, the RSMT does not tell which L-shape to use, and we can either go horizontally or vertically first. Therefore, the entire rectangular region that covers the two points will be considered as part of the routing region to provide more flexibility.

In this way, a smaller routing region like Figure 3.3(b) will be generated. It is worth noting that for the coming fine-grained maze routing, the region will be expanded vertically to form a 3D subgraph, where the routing layer range will be initially determined by the min-routing-layer and the pin locations of the net. Based on that, we will expand the layer range both upwards and downwards to increase the routing flexibility. As shown in Figure 3.3(c), with the help of route guides, a fine-grained routing solution with short wire length can be efficiently retrieved in a much smaller solution space.

(b) Cost Scheme: In the fine-grained maze routing, we assign a cost to each GCell, taking into account congestion. The cost function $f(v)$ for a GCell v is described by,

$$f(v) = 1 + con(v) + op(v), \tag{3.2a}$$

$$con(v) = \frac{\alpha}{1 + exp(cap(v))}, \tag{3.2b}$$

$$op(v) = inf \times \mathbb{1}(cap(v) \leq 0), \tag{3.2c}$$

where $\mathbb{1} \in \{0, 1\}$ is an indicator function. We can see that $f(v)$ consists of three parts: the wire length cost (which is always one), the congestion cost, and the overflow penalty. The congestion cost $con(v)$ is designed to (i) increase rapidly when the capacity is close to zero and (ii) incur little cost when the routing resource is abundant. Here α is a constant corresponding to the weight of the congestion cost. Lastly, if $cap(v) \leq 0$, that is, overflow has occurred or adding just one more demand will cause overflow, a large penalty will be invoked by $op(v)$.

(c) Greedy Rip-up and Reroute: Based on the RSMT route guides and the cost scheme, a greedy rip-up and reroute (RRR) is implemented to improve an existing routing solution in terms of congestion and wire length. The greedy RRR process

is described as follows. Nets will first be sorted by the HPWL in ascending order. To guarantee an overflow-free solution, nets will be rerouted one by one. The new routing solution of a net will be accepted if no overflow occurs and the corresponding wire length is shorter than the old one.

3.2.3 Cell Movement with A* Search-Based Partial Rerouting

In this section, a multi-threaded cell movement scheme is proposed. In this process, we need to identify the cells to be moved and their potential destinations. To achieve this, we devised a lookup table-based approach to estimate quickly wire length gain after cell relocation utilizing the existing net routing segments, and an A* search-based partial rerouting algorithm for fast routing tree reconstruction.

(a) Lookup Table-based Wire Length Estimation: We first explain how routed wire length can be accurately measured by a lookup table-based approach under the min-routing-layer and preferred routing direction constraints. For two points $p_1 = (r_1, c_1, l_1)$ and $p_2 = (r_2, c_2, l_2)$, without any constraint, as shown in Figure 3.4(a), the routing distance between them can roughly be estimated as the summation of $|r_1 - r_2|$, $|c_1 - c_2|$, and $|l_1 - l_2|$.

However, when the two routing layer-related constraints are involved, the calculation becomes less straightforward. Figure 3.4(b) shows one shortest routing path to connect a two-pin net when $mrl = 2$. Starting from A, the route needs to reach M2 first because of the min-routing-layer constraint. Vertical in-layer routing will then be conducted. Due to the existence of preferred routing directions, there must be an additional pair of up-and-down movements in order to conduct horizontal routing on M3. Calculation of such minimum routing distance may look complicated and can be affected by various factors.

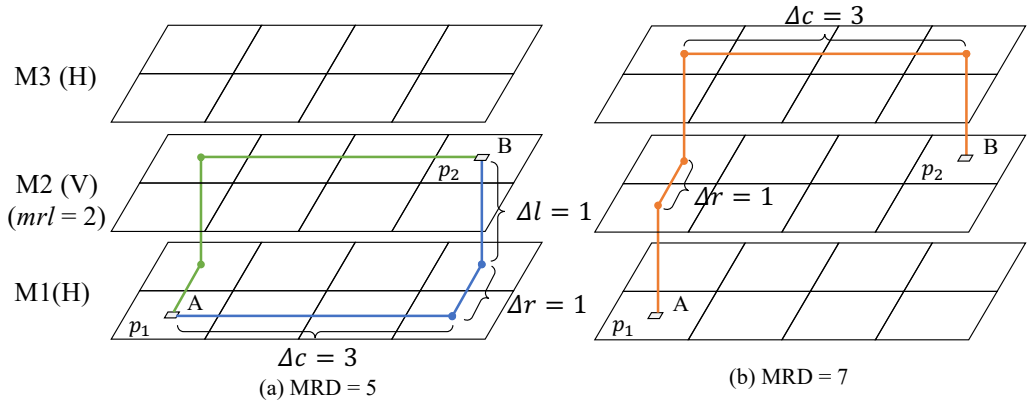


Figure 3.4: *Illustration for the minimum routing distance (MRD) calculation. (a) Without any constraints, MRD can be estimated as the summation of* $|\Delta r| + |\Delta c| + |\Delta l|$*. (b) With the preferred routing direction and mrl,* $|\Delta r| + |\Delta c|$ *can be one invariant in the complicated MRD calculation.*

Despite that, the routing distance can still be estimated efficiently with the help of a pre-calculated lookup table. We denote the *minimum routing distance* (MRD) needed to connect two points, p_1 and p_2, with a minimum routing layer *mrl* by a function $mrd(p_1, p_2, mrl)$. In Figure 3.4(b), we can see that when calculating $mrd(p_1, p_2, mrl)$, $|\Delta r| = |r_1 - r_2|$ and $|\Delta c| = |c_1 - c_2|$ always contribute to the total routing distance. It is the capturing of via usage under different situations that makes the routing distance estimation complicated. An accurate estimation of via usage can be obtained with the following information: (i) whether the two points are on the same row, (ii) whether the two points are on the same column, (iii) the layer of p_1, (iv) the layer of p_2, (v) the *mrl* value, and (vi) the preferred routing direction for each layer in the current design. A via-usage lookup table $vlut \in \mathbb{N}^{2\times2\times L\times L\times L}$ can then be pre-calculated in $O(L^3)$ time by enumerating all the possibilities. Since L can be at most 16 in all the given cases, the space and preprocessing overhead is negligible.

With the above definitions, the formulation of *mrd* is as follows,

$$mrd(p_1, p_2, mrl) = |\Delta r| + |\Delta c| + vlut(zero(\Delta r), zero(\Delta c), l_1, l_2, mrl), \tag{3.3a}$$

$$zero(x) = \begin{cases} 1 & \text{, if } x = 0, \\ 0 & \text{, if } x \neq 0. \end{cases} \tag{3.3b}$$

Back to the case in Figure 3.4(b), the *mrd* value can be quickly computed as $|1-2| + |1-4| + vlut(0,0,1,2,2) = 1+3+3$.

(b) Cell Movement Gain Estimation : With accurate point-to-point wire length estimation, we further discuss the cell movement gain estimation that utilizes the existing net routing topologies, which will be critical for selecting a set of good candidate destinations for the placement optimization purpose.

For a multi-pin net n, let C_n be the set of cells connected by n and $G(n)$ be the set of visited GCells in the current routing solution. Under the circumstance that one of its cells $i \in C_n$ is moved, we denote the *remaining segments* (RS) as $rs(n,i)$, which is the smallest subset of $G(n)$ such that all the cells in $C_n \setminus i$ are still connected without violating any routing constraints.Figure 3.5 gives an illustration for the remaining segments when cell C is moved. Further, we denote the removed routing segments due to the cell movement as $rm(n,i) = G(n) \setminus rs(n,i)$. In this way, the rest of the net $rs(n,i)$ can still form a single connected component, and the routing tree can be easily reconstructed by connecting the moved cell to any one of the GCells in $rs(n,i)$. Note that both $rs(n,i)$ and $rm(n,i)$ can be quickly retrieved by a search on the routing tree, starting from the GCell containing the pin of the moved cell i.

Next we (i) conduct the cell movement gain estimation and (ii) select candidate

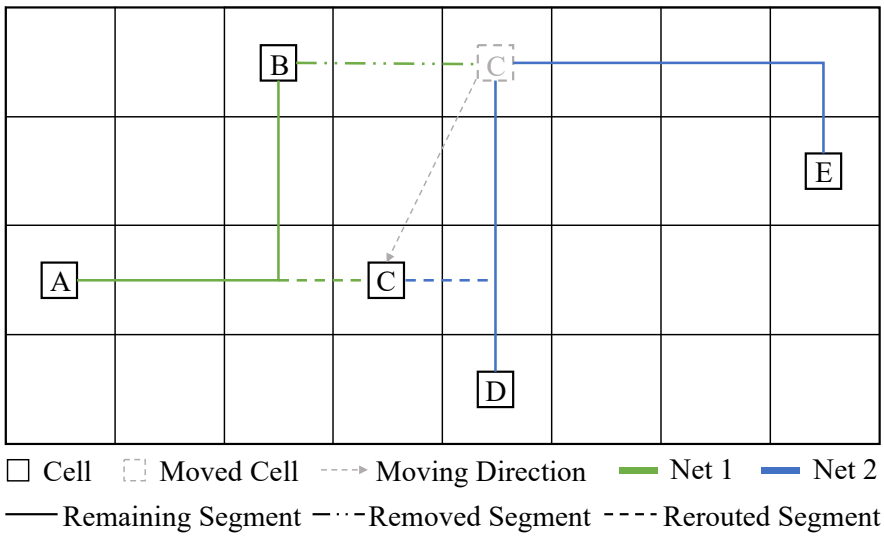


Figure 3.5: *Illustration for the remaining segments and the removed segments. Note that the remaining segments of Net 1 and Net 2 both form a single connected component.*

destinations for each cell. Consider a single cell i, let N_i be the set of nets associated with cell i. As Algorithm 1 shows, for gain estimation, we first get a rectangular *candidate region* (denoted as *candiRegion* in line 1) where the wire length gains will be calculated for each location (r, c) in *candiRegion* and stored in a gain map denoted as $gainMap$. The gain map will be initialized to zero at the beginning (line 2) and updated from time to time. We will later discuss how this candidate region can be obtained.

As line 3 shows, we will iterate through all the nets in N_i and accumulate the cell movement gains for each location (r, c) in $gainMap$ (line 25). When handling a net n, we first find the pin in n that belongs to cell i to get the layer information of the point to be connected (line 4-5). With the remaining segments $rs(n, i)$ and removed segments $rm(n, i)$ retrieved from the routing tree of n, the wire length reduction wl_{rm} due to the removal of cell i will simply be the size of $rm(n, i)$ (line 6-8). In line 9-11, the boolean variable *sameLoc* indicates if all the GCells in the remaining segments have the same 2D coordinate, which can make a significant difference in

Algorithm 1 Cell Movement Gain Estimation

Input: Cell i, Set of nets N_i associated with cell i.

Output: A map $gainMap$ recording the estimated gains when cell i is moved to different locations.

1: $candiRegion \leftarrow getCandidateRegion(i)$;
2: Initialize each entry of $gainMap$ to 0;
3: **foreach** $n \in N_i$ **do**
4: Let $pin_n \in n$ be a pin of cell i;
5: $l \leftarrow getLayer(pin_n)$;
6: $rs(n,i) \leftarrow getRemainingSegments(n,i)$;
7: $rm(n,i) \leftarrow getRemovedSegments(n,i)$;
8: $wl_{rm} \leftarrow |rm(n,i)|$; ▷ Get removed wire length.
9: $sameLoc \leftarrow true$;
10: **if** $rs(n,i)$ contains GCells with different 2D coordinates **then**
11: $sameLoc \leftarrow false$; ▷ For higher estimation accuracy.
12: **foreach** $dest = (r,c) \in candiRegion$ **do**
13: Let $p = (r,c,l)$;
14: $cost_{min} \leftarrow \infty$;
15: **foreach** GCell $p' = (r',c',l') \in rs(n,i)$ **do**
16: **if** $r = r'$ **and** $c = c'$ **and** $sameLoc = false$ **then**
17: **if** $l < mrl(n)$ **and** $l' < mrl(n)$ **then**
18: $cost_{cur} \leftarrow max(l' - l, 0)$;
19: $cost_{min} \leftarrow min(cost_{min}, cost_{cur})$;
20: **else**
21: $cost_{min} \leftarrow min(cost_{min}, mrd(p, p', mrl(n))$;
22: **else**
23: $cost_{min} \leftarrow min(cost_{min}, mrd(p, p', mrl(n))$;
24: $wl_{gain} \leftarrow wl_{rm} - cost_{min}$;
25: $gainMap[r][c] \leftarrow gainMap[r][c] + wl_{gain}$;
26: **return**

the later estimation process.

In line 12-24, we calculate the wire length gain of net n when the cell is moved to different positions in $candiRegion$. Consider a possible destination (r,c), the corresponding point to be connected back to $rs(n,i)$ will be $p = (r,c,l)$ (line 13). We then iterate through all the GCells $p' \in rs(n,i)$ to find the minimum wire length needed for the reconnection (line 15-23). In most cases, the minimum value can be

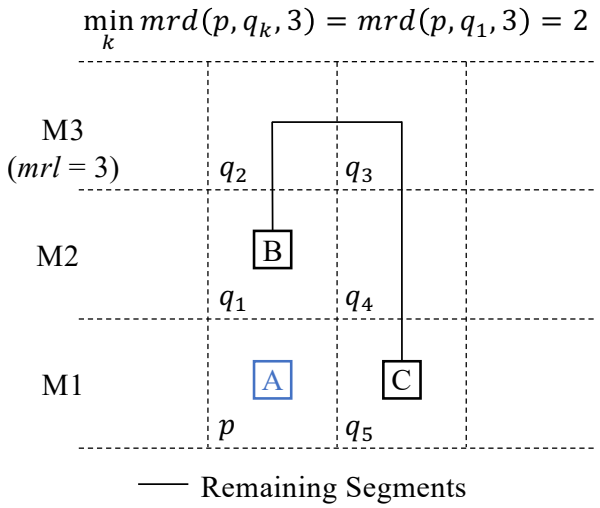


Figure 3.6: *Illustration for the special case handling for accuracy improvement under the presence of min-routing-layer.*

easily updated with the *mrd* function, which is based on a lookup table (line 23). However, some special case handling is needed for higher estimation accuracy (line 16-19) as explained below.

An example is shown in Figure 3.6. In this example, we consider moving cell A to GCell p and try to estimate the minimum routing distance for the reconnection. This is a case that the destination position overlaps with part of the remaining segments in 2D. In the presence of min-routing-layer M3, if we simply measure the *mrd* between p and each q_k in the remaining segments, $mrd(p, q_1, 3) = 2$ will be considered as the minimum wire length needed to connect pin A with the remaining parts. However, the actual additional wire length needed is 1 only, which corresponds to the usage of GCell p. The over-estimation by $mrd(p, q_1, 3)$ is resulted from the ignorance of the fact that GCell q_2 has already been included in the remaining segments when we are considering the (p, q_1) pair. The checkings and steps in line 16-19 can help to improve the estimation accuracy under similar situations, which is essential as the min-routing-layer can be as high as ten in some

Algorithm 2 Candidate Destination Selection

Input: Cell i, Gain estimation map *gainMap* for cell i.
Output: Candidate destinations with movement gains for cell i.

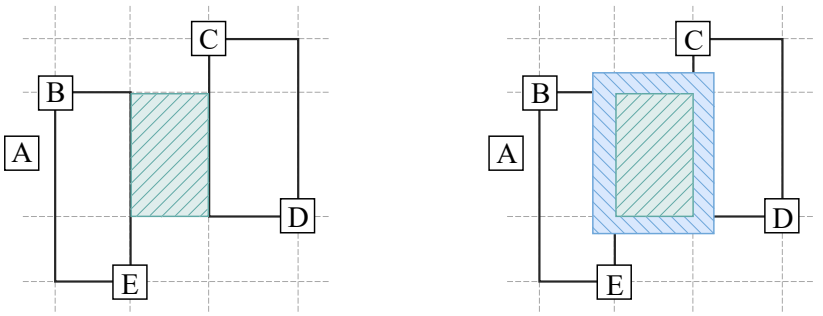
```
1: candiDest ← {};                          ▷ A collection of (gain, destination) pairs.
2: candiRegion ← getCandidateRegion(i);
3: foreach dest = (r, c) ∈ candiRegion do
4:     if gainMap[r][c] > 0 then
5:         candiDest.append(gain, dest);
6: Sort candiDest by gain in descending order;
7: return
```

benchmarks.

After cell movement gain estimation and updates to gain map, we can conduct candidate destination selection as shown in Algorithm 2. For each position in the candidate region, if the estimated wire length gain is greater than zero, it will be added to the candidate destination collection *candiDest* along with its gain (line 3-6). Sorting will be conducted based on the gains such that positions with higher potential of reducing wire length will be tried first.

(c) Candidates Pruning by Median Box: At the beginning of this cell movement gain estimation stage, we will locate the candidate region. For efficiency, it is not good to take the whole $R \times C$ layout as the candidate region. Consider current cell i, let N_i be the set of nets connecting to it, and C_i be the set of cells connected to i. An intuitive idea will be to consider the smallest bounding box that covers all the cells in C_i and cell i itself as the region, which will be referred to as the *cell bounding box*. However, with this approach, the bounding box can still be large when C_i spans a wide area, which dramatically slows down the estimation process.

Therefore, we introduce a median box-based approach to generate candidate region, which effectively prunes the total number of candidates without affecting the solution quality. In the proposed approach, an *expanded optimal region* will



(a) Optimal Region (b) Expanded Optimal Region

Figure 3.7: *Illustration of the expanded optimal region for cell A, which will be taken as the candidate region.*

be taken as a candidate region, which is defined based on a concept called *optimal region (OR)*. We define optimal region for cell i (OR_i) as the set of locations that minimize $\sum_{n \in N_i} hpwl(n)$ when cell i is being moved. Here $hpwl(n)$ refers to the half-perimeter wire length of net n. As mentioned in [102], OR_i can be computed based on the median of the bounding box boundaries of the nets in N_i. For a net $n \in N_i$, we denote the lower-left and upper-right corner of its bounding box with cell i removed as $(r_1(n),\ c_1(n))$ and $(r_2(n),\ c_2(n))$ respectively. Let $Rs = [r_1(n_1), r_2(n_1), r_1(n_2), r_2(n_2), ..., r_1(n_{|N_i|}), r_2(n_{|N_i|})]$ be an array of all these row indices. Similarly, we have an array Cs storing the column indices. OR_i will then be the rectangular region with (r_1^{opt}, c_1^{opt}) as its lower-left corner and (r_2^{opt}, c_2^{opt}) as its upper-right corner, where r_1^{opt}, r_2^{opt} are the medians of Rs and c_1^{opt}, c_2^{opt} are the medians of Cs. Note that both Rs and Cs will have an even number of elements. It is possible that r_1^{opt} equals r_2^{opt} or c_1^{opt} equals c_2^{opt}. In that case, OR_i will be a single line or even a single point. Figure 3.7(a) shows the optimal region for a single cell A in a case that there are only two nets {A, B, E}, {A, C, D} containing A. With cell A excluded, the optimal region (green region) can be calculated using the bounding boxes of the two nets. However, when all the positions in OR_i are congested, moving

the cell to a location nearby may still lead to a considerable amount of wire length reduction. As shown in Figure 3.7(b), by expanding the optimal region by a margin, an expanded optimal region is obtained and will serve as the candidate region. In Section 3.3, we will show the effectiveness of this pruning technique and explore a suitable width for the candidate margin.

(d) A* Search-Based Partial Rerouting: With a set of good cell movement candidate destinations, routing is needed to rebuild net connection and verify if the cell can be successfully moved to the new location, that is, no overflow is caused and the actual wire length gain through this relocation is large enough. In alignment with the proposed estimation scheme, and to quickly connect a cell back to the remaining segments of its associated nets, a multi-source single-target A* search-based partial rerouting algorithm is proposed (Algorithm 3). Consider a net n, given its remaining segments rs, and a pin to be reconnected $p = (r, c, l)$, we try to find a least-cost path from rs to pin p. In the algorithm, we assume that $p \notin rs$, or else the remaining segments can be directly used as the new routing solution. As two critical elements in an A* search, the routing cost and the heuristic cost of a GCell v are modeled by $f(v)$ from Equation (3.2) and $mrd(v, p, mrl(n))$ from Equation (3.3a) respectively.

In the proposed A* search, we maintain a map *cameFrom* for routing path traceback, a map *rScore* for the minimum routing cost from any point to rs, and a map $fScore$ for the heuristic score based on function f and mrd (line 1-4). We treat all the GCells in rs as sources, updating the corresponding entries in *rScore* and $fScore$ (line 5-8), and conduct a typical A* search (line 9-22). Once the pin p is found, we construct the routing solution following the *cameFrom* map (line 11-13). Routing failure will be reported if the pin cannot be reached due to the preferred routing direction or min-routing-layer constraints (line 23-24). Since the

Algorithm 3 A* Search-Based Partial Rerouting

Input: Net to be partially rerouted n, Remaining segments rs, To be connected pin $p = (r, c, l)$.
Output: A new routing solution for net n.

1: $openSet \leftarrow \{\}$; ▷ A set of discovered nodes to be expanded.
2: $cameFrom \leftarrow$ empty map; ▷ For routing path tracing.
3: $rScore \leftarrow$ map with all ∞; ▷ Minimum routing cost from sources.
4: $fScore \leftarrow$ map with all ∞; ▷ Heuristic score based on f and mrd.
5: **foreach** GCell $p' \in rs$ **do**
6: $openSet.add(p')$; ▷ Treat all GCells in rs as sources.
7: $rScore(p') \leftarrow 0$;
8: $fScore(p') \leftarrow mrd(p', p, mrl(n))$; ▷ mrd for estimation.
9: **while** $openSet$ is not empty **do**
10: Let $u \in openSet$ be the node with the lowest $fScore$ value;
11: **if** $u = p$ **then** ▷ The goal is reached.
12: Construct the routing solution with $cameFrom$;
13: **return**
14: $openSet.remove(u)$;
15: **foreach** neighbor v of u **do**
16: $rScore_{new} \leftarrow rScore(u) + f(v)$; ▷ $f(v)$: cost for a GCell.
17: **if** $rScore_{new} < rScore(v)$ **then**
18: $cameFrom(v) \leftarrow u$;
19: $rScore(v) \leftarrow rScore_{new}$; ▷ Update the minimum cost.
20: $fScore(v) \leftarrow rScore(v) + mrd(v, p, mrl(n))$;
21: **if** $v \notin openSet$ **then**
22: $openSet.add(v)$;
23: Report routing failure;
24: **return**

given placement has been optimized globally, cells will not be moved far away from its original location in most of the cases. Therefore, the proposed path searching algorithm can be highly efficient in this routing with cell movement context.

With the proposed A* search-based partial rerouting algorithm, the overall flow of a cell movement is described in Algorithm 4. Before we move a cell i, the total wire length of the previous routing solution will be recorded first (line 1). We then rip up the cell from the current placement (line 2). To reclaim part of the resource used by the previous routing solution and prepare for the future partial rerouting,

Algorithm 4 Single Cell Movement

Input: Cell i, Set of nets N_i associated with cell i, Sorted candidate destinations with movement gains $candiDest_i$, Gain threshold τ.
Output: New placement for cell i and update routing solutions.

1: $wl_{old} \leftarrow \sum_{n \in N_i} wl(n)$; ▷ Original total wire length.
2: Rip up cell i from current placement result;
3: **foreach** net $n \in N_i$ **do**
4: $rs(n,i) \leftarrow getRemainingSegments(n,i)$;
5: $rm(n,i) \leftarrow getRemovedSegments(n,i)$;
6: Rip up the routing resource usage by $rm(n,i)$;
7: $moveSuccess \leftarrow false$;
8: **foreach** $(gain, dest) \in candiDest_i$ **do**
9: **if** $gain < \tau$ **then**
10: Break out of the loop;
11: Move cell i to $dest$;
12: Perform partial rerouting for all $n \in N_i$, store the results in N_i';
13: $wl_{new} \leftarrow \sum_{n' \in N_i'} wl(n')$;
14: $gain_{wl} \leftarrow wl_{old} - wl_{new}$;
15: **if** $gain_{wl} \geq \tau$ and no overflow occurs **then**
16: Accept the cell movement and update routing solution;
17: $moveSuccess \leftarrow true$;
18: Break;
19: **if** $moveSuccess = false$ **then**
20: Put cell i back to its previous location;
21: Recover old routing solution for all $n \in N_i$;
22: **return**

the remaining and removed segments will be retrieved for each net associated with cell i (line 3-6). After that, we will consider every position $dest$ in the pre-calculated candidate destinations $candiDest_i$, starting from the one with the largest $gain$ (line 8-18). After partial rerouting, the actual gain in wire length $gain_{wl}$ will be calculated as the total wire length change before and after the cell movement (line 12-14). A cell movement will be accepted when there is no overflow and $gain_{wl}$ is greater than or equal to the gain threshold (line 15-18). If the cell movement fails for all candidate destinations, the cell will be put back to its original position with related routing solution restored (line 19-21). Note that in line 9-10, a pruning based

on the estimated gain is conducted to speed up the process without affecting the solution quality. As the cell movement estimation scheme does not consider routing congestion, $gain$ is an upper bound of the actual wire length gain $gain_{wl}$. Therefore, if $gain < \tau$, the cell movement will be rejected immediately. Since the $candiDest_i$ is sorted by $gain$, there is no need to consider the rest of the destinations if the current candidate already has an estimated $gain$ less than τ.

For an independent batch of cells, the movement step can be parallelized by dividing the cells into several mini-batches in such a way that cells in the same batch do not overlap in terms of placement and routing region.

(e) Selective Net Rerouting: In Figure 3.5, we can see that in the original routing solution of Net 2, a C-shaped detour was made to connect cell C. After the relocation of cell C, this unnecessary detour, still exists in the new routing solution as it was treated as parts of the remaining segments. To handle such cases, as shown in Figure 3.2, we will conduct one round of *selective net rerouting*, which only selects the nets associated with the successfully moved cells in the previous round of cell movement, and performs greedy RRR (end of Section 3.2.2) for them. In this way, new routing topologies can be generated if the existing ones can be further improved.

3.2.4 Post-processing with Greedy Cell Put-back

After the multi-threaded cell movement process, several post-processing techniques are applied to further reduce the total routed wire length while ensuring a legal output solution. In the proposed flow, we allow more than $MaxCellMove$ cells to be moved. When the total number of moved cells exceeds $MaxCellMove$, a greedy cell put-back process will be applied to carefully move cells that bring fewer gains back

to their original locations without causing overflow. This is done as follows. We will start with a gain threshold (-1 in our implementation), and try to put each moved cell back to its initial position. Note that the gain threshold is negative because the wire length is expected to increase in this put-back process. However, if the wire length increases by too much, that is, the wire length gain is smaller than the gain threshold, or overflow occurs, we will not move a cell back to its original position to avoid degrading the solution quality for too much.

After iterating through all the cells once, the gain threshold will be reduced to allow more cells to move back in the next iteration. This process will stop once the total number of moved cells is no greater than *MaxCellMove*. After this cell put-back process, several rounds of wire length-driven rerouting, which is essentially the aforementioned greedy RRR with little congestion cost, will be performed to further reduce the total wire length.

3.3 Experimental Results

Our P&R co-optimization engine is implemented in the C++ language and all the experiments were conducted on a Linux machine with 2.90 GHz Intel Xeon CPU and 756 GB memory. Consistent with the contest, eight threads are used by default. We empirically test the proposed flow on the ICCAD 2020 contest benchmark suites [51] and compare our results with the top-3 winners. Ablation studies are also conducted to show the effectiveness of our proposed techniques with respect to (i) solution quality and (ii) runtime.

Table 3.1: *Benchmark Statistics*

Case ID	#Cells	#Nets	*MaxCellMove*	#GCells ($Row \times Col \times L$)
case1	8	6	2	5× 5×3
case2	6	6	3	4×4×3
case3	2735	2644	820	27×33×7
case4	204206	179996	61261	277×277×12
case5	96682	92546	29004	104×103×16
case6	352269	332080	105680	237×236×16
case3B	2604	2563	781	29×29×7
case4B	207347	183137	62204	277×277×12
case5B	96689	92559	29006	104×103×16
case6B	352234	332045	105670	237×236×16

3.3.1 Comparison with the Top-3 Winners

The benchmark consists of ten cases, including two toy cases (case1, case2). In Table 3.1, statistics of each cases are shown, including the number of cells, the number of nets, the maximum number of cells that can be moved, and the size of the layout.

Quantitative results are presented in Table 3.2. Column "Score", as formulated in Equation (3.1), measures the wire length improvement between the given input solution and the final output solution. Column "R_{diff}", defined as $R_{diff} = \frac{Score}{TWL_{input}} = 1 - \frac{TWL_{output}}{TWL_{input}}$, denotes the total wire length reduction rate. From the table, we can observe that our proposed co-optimizer can effectively reduce the total routed wire length based on a given P&R solution. Compared with the ICCAD 2020 contest winners, our co-optimizer stands out with better scores and much shorter runtime in all the cases. Specifically, compared with the first place, we achieves 0.9% better scores with 87% less runtime on average. In the contest setting, this can be considered as a significant gap, given that the differences in score between the first place and the other two top-3 participants are only 0.8% and 1.0% respectively

Table 3.2: *Experimental Results on the ICCAD 2020 Benchmark*

Benchmarks		1st Place Team		2nd Place Team		3rd Place Team		Ours (Starfish)		
Case ID	TWL_{input}	Score	RT (s)	Score	RT (s)	Score	RT (s)	Score	RT (s)	R_{diff}
case3	32600	11425	34	11428	4	11557	111	**11610**	2	0.356
case4	4680681	2046811	2221	2048105	804	2037598	3441	**2064790**	260	0.441
case5	1763627	695219	903	685173	183	682963	1213	**699626**	111	0.397
case6	7188481	2721274	3171	2687926	2217	2656320	3511	**2737028**	684	0.381
case3B	29748	11237	31	11073	4	11289	206	**11327**	1	0.381
case4B	4886698	2182574	2299	2180172	786	2167411	3371	**2200820**	284	0.450
case5B	1721530	664347	886	654797	237	654183	1226	**668351**	103	0.388
case6B	7340802	2748097	3406	2722222	2801	2668052	3514	**2785061**	932	0.379
Avg. ratio	-	1.000	1.000	0.992	0.368	0.990	2.224	1.009	0.133	0.397

* The score and runtime statistics of the top-3 winners are provided by the contest organizer with Intel Xeon E7-4820 CPU (2.00 GHz, 8 cores) and 128 GB memory. TWL_{input} denotes the total wire length in the given input solution. RT is short for runtime. $R_{diff} = Score/TWL_{input}$. The average ratio for score is obtained by taking the average of normalized scores against the 1st place, same for the runtime.

on average. Moreover, results from column "R_{diff}" shows that with the proposed routing with cell movement scheme, our co-optimizer can achieve 39.7% total wire length reduction on average.

3.3.2 Effectiveness of Wire Length Reduction

To demonstrate the effectiveness of our proposed techniques for quality improvement, in Figure 3.8, we show the total wire length reduction after each stage of our flow, including the initial solution refinement, the initial rerouting, the multi-threaded cell movement, the greedy cell put-back, and the wire length-driven rerouting. Note that both the refinement and the initial rerouting can effectively reduce the total wire length. Cell movements are then performed to co-optimize placement and routing. After cell put-back, the score will drop since the previous cell movement stage overshot the maximum number of cells that can be moved,

and this is a stage to correct it. However, the drop is very minimal because of the techniques applied in the put-back process. Lastly, the total wire length is further reduced by the wire length-driven rerouting.

In the overall flow of Starfish shown in Figure 3.2, a gain threshold τ is used to maintain an implicit cell ordering. We study how the total score (summation of scores from all benchmarks except for the two toy cases) will change with different initialization of τ. The result is shown in Figure 3.9. We can see that a larger τ will give better result, but a larger τ will also lead to a longer runtime to converge. In our implementation, we initialize τ as 15.

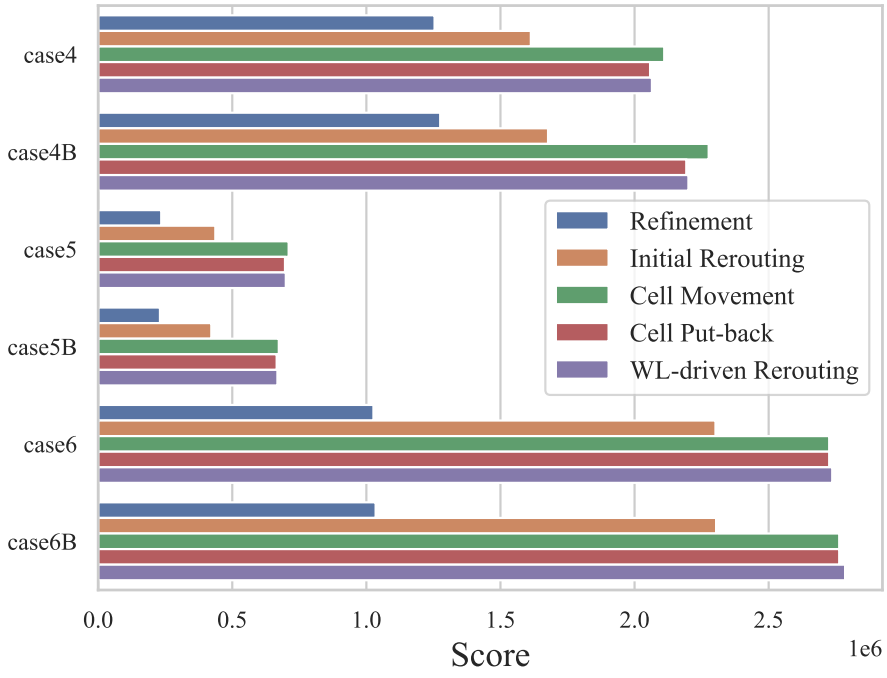


Figure 3.8: *Illustration for the wire length reduction after each stage in our flow. For case6 and case6B, since the cell movement quota is not used up in the end, the cell put-back does not affect the score.*

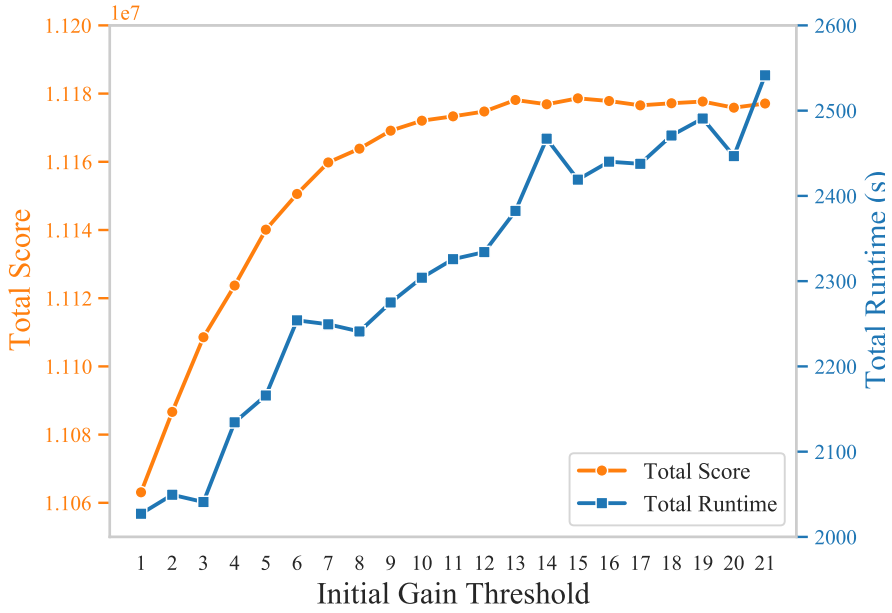


Figure 3.9: *The total score here refers to the summation of scores from all the cases except for the two toy cases, same for the runtime. In general, both the total score and runtime increase with a larger initial gain threshold.*

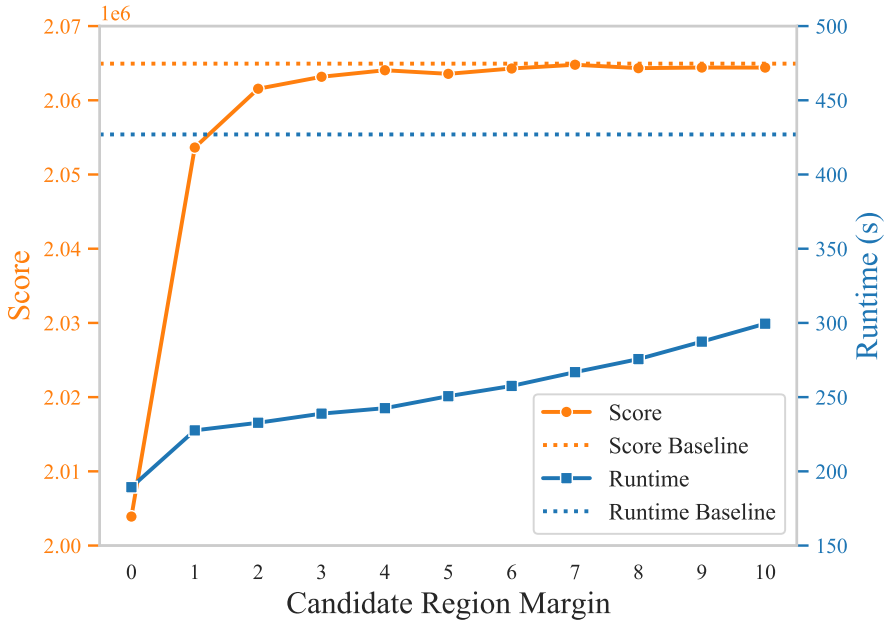


Figure 3.10: *The impact of the candidate region margin on the total wire length reduction (score) and runtime in case4. As a baseline, the cell bounding box will be used as the candidate region.*

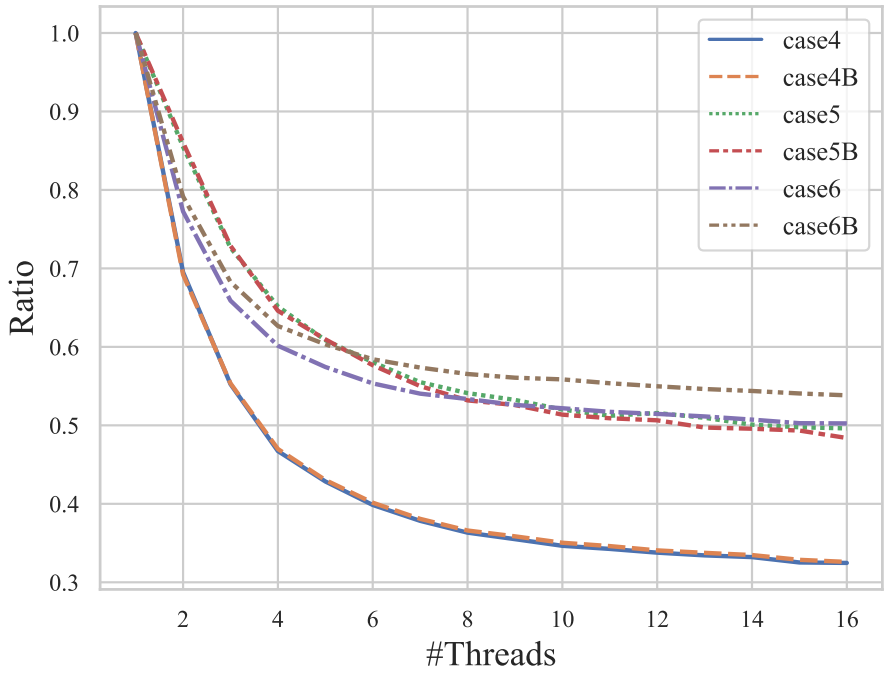


Figure 3.11: *Speed-up by multi-threading.*

3.3.3 Effectiveness of Runtime Reduction

In Section 3.2.3, a median box-based approach is proposed for efficient solution space pruning. In comparison with the intuitive idea of taking the cell bounding box, which covers a cell and all its connected cells, as the candidate region, we show the effectiveness of our pruning technique with different levels of expanding the optimal region (called the candidate region margin) in Figure 3.10, where case4 is used for demonstration. Here, the baseline uses the cell bounding box as the candidate region, which is supposed to be large enough to give very good result. However, a large candidate region will also lead to a long runtime. To reduce runtime, a median box with an expanded margin is used in our approach. Comparing with the baseline, our proposed approach can effectively reduce the total runtime while achieving a very similar quality. For example, with a margin close to 4, a result with

decent quality can be obtained with an over 40% runtime reduction. Note that the curves are similar in other benchmarks. In our implementation, seven is chosen as the margin for better quality and runtime trade-off.

For parallelism, multi-threading is applied in greedy RRR, cell movement gain estimation, and cell movements. We show the effectiveness of these multi-threading strategies in Figure 3.11 by changing the number of threads and measure the corresponding runtime improvement. The runtime ratio is obtained by comparing with the performance of the proposed framework in a single-threaded execution.

3.4 Summary

In this chapter, we propose Starfish, an efficient P&R co-optimization engine, to bridge the gap between placement and routing. We demonstrate that the proposed flow can effectively reduce the total routed wire length by routing with cell movement. Experimental results on the ICCAD 2020 benchmark suites show that, with an accurate cell movement gain estimation scheme and several efficient routing algorithms, Starfish is able to achieve an average total wire length reduction of 39.7% compared with the initial P&R solution. Besides, our co-optimizer outperforms all the contestants with better solution quality and much shorter runtime. In particular, compared with the champion of the contest, Starfish can achieve 0.9% better scores with 87% less runtime.

Chapter 4

Fast Pin Access Analysis for High-Quality Detailed Routing

In this chapter, we propose FastPass, a boolean satisfiability (SAT)-based pin access analysis framework (PAAF) for efficient and correct-by-construction pin access scheme generation, which enables high-quality detailed routing. Our main contributions are summarized as follows.

- A pre-processing flow that includes instance pattern-based analysis and inter-instance analysis is proposed. This flow aims to efficiently generate pin access route candidates for each pin while ensuring adherence to design rules. Additionally, inter-route conflicts are pre-computed as part of this process.
- A sweep line-style algorithm is developed for quick design rule checking and inter-route conflict detection, enabling the highly-efficient pre-processing flow.
- An SAT formulation for the conflict-free route selection problem is proposed and incremental SAT solving is adopted to achieve a pin access scheme that is not only DRC-clean but also optimized towards custom objectives.

- FastPass is further integrated into Dr. CU to demonstrate its effectiveness.
- Experimental results on the ISPD 2018 Initial Detailed Routing Contest benchmark suite [87] show that FastPass can produce DRC-clean pin access schemes while being an order of magnitude faster than the state-of-the-art PAAF [60], which will be referred to as TOP in the rest of this chapter. Besides, with the integration of FastPass, Dr. CU is able to produce much better results with 87.5% less short area and 72.4% fewer DRC violations.

The rest of the chapter is organized as follows. In Section 4.1, we introduce the problem formulation of pin access anaylsis along with related concepts. In Section 4.2, we explain the algorithms in FastPass and the techniques employed to integrate FastPass into Dr. CU. Experiments that cover both pin access analysis and detailed routing results will be discussed in Section 4.3.

4.1 Preliminaries

4.1.1 Problem Formulation

We formally define the problem of DRC-clean pin access scheme generation as follows.

Given a design (with tracks, cell instances, nets, etc.), find a routing scheme such that each net pin is connected to a nearby grid point using *routes* (a collection of wires and vias), such that no routes will result in DRC violations with fixed metals (i.e., pin shapes or routing blockages) or with each other.

Figure 4.1a shows a toy example with three simplified cell instances, and Figure 4.1b gives an example of a DRC-clean pin access scheme. There are several factors that make this problem non-trivial. (i) The complex pin shapes must be

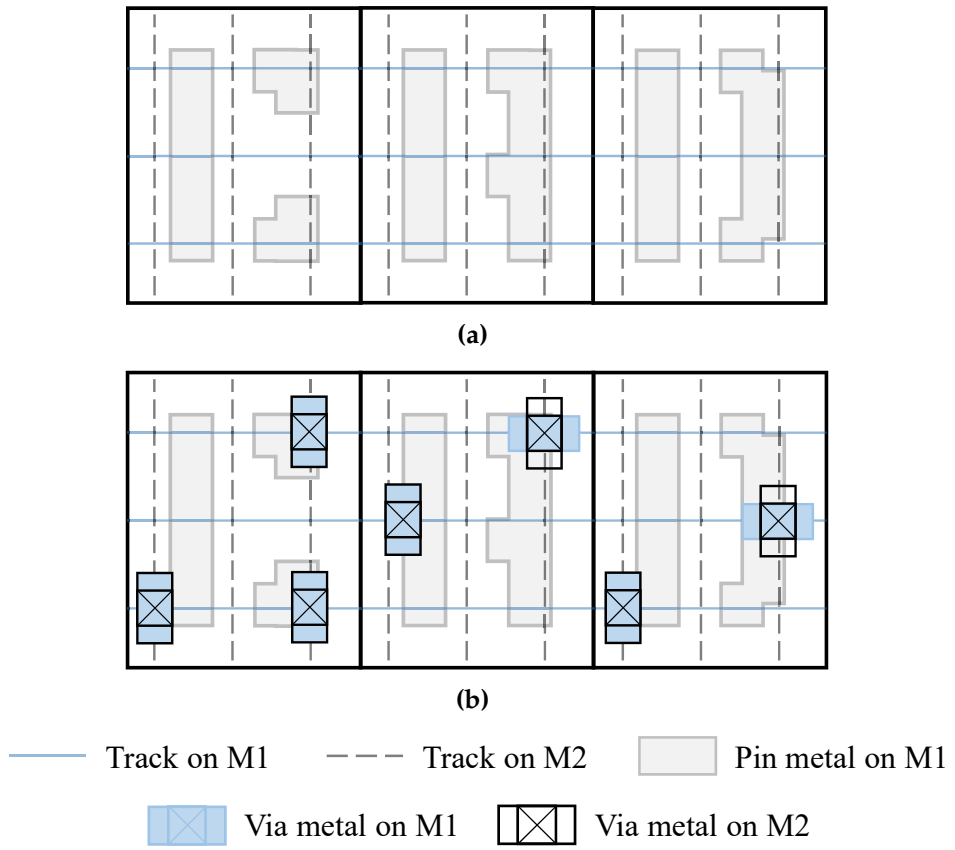


Figure 4.1: *An Example Pin Access Problem.*

analyzed with various design rules to rule out routes that have DRC violations with them. (ii) The routes for different pins must be decided simultaneously to avoid DRC conflicts between them.

4.1.2 Instance Patterns

Following a similar approach as [96] and [60], we group the cell instances into *instance patterns* to prevent redundant analysis of instances with identical patterns. Two instances are categorized under the same pattern when they share the same master cell type, orientation, and track offset (which refers to the distance from the first in-cell track to the left boundary of the instance). For instance, in Figure 4.2a,

the instance is considered to have the same pattern as the rightmost instance in Figure 4.1. However, the instances depicted in Figure 4.2b and Figure 4.2c do not share the same pattern, differing in orientation and track offset, respectively.

4.1.3 Design Rule Checking

During pin access analysis, we mainly consider four types of DRC violations described in the ISPD 2018 contest [87], which can be summarized as follows:

(1) *Short*: A via metal or wire metal is not allowed to overlap with any other object (i.e., metals, blockages, pin shapes) unless both objects are from the same net.

(2) *Parallel run length (PRL) spacing*: A minimum spacing is required for two metal objects based on their maximum width and the length they run parallel to each other. Note that with non-positive PRL, a minimum spacing depending on the maximum width will still be required.

(3) *End-of-line (EOL) spacing*: A polygon edge shorter than a threshold *eolWidth* is considered an EOL edge. This rule specifies a region from an EOL edge to the exterior of the polygon where no objects should exist.

(4) *Cut spacing*: Vias across the same cut layer should have a minimum spacing between their cut shapes.

In the context of pin access, DRC violations primarily arise from two sources, (i) violations between a routed segment (wires or vias) and a fixed metal and (ii) violations between different routed segments. Figure 4.3a shows a PRL spacing violation between a routed via and the metal of a different pin on M1. Recall that the required spacing depends on the maximum width between two objects. In this case, the via metal and the upper pin collectively form an object of increased

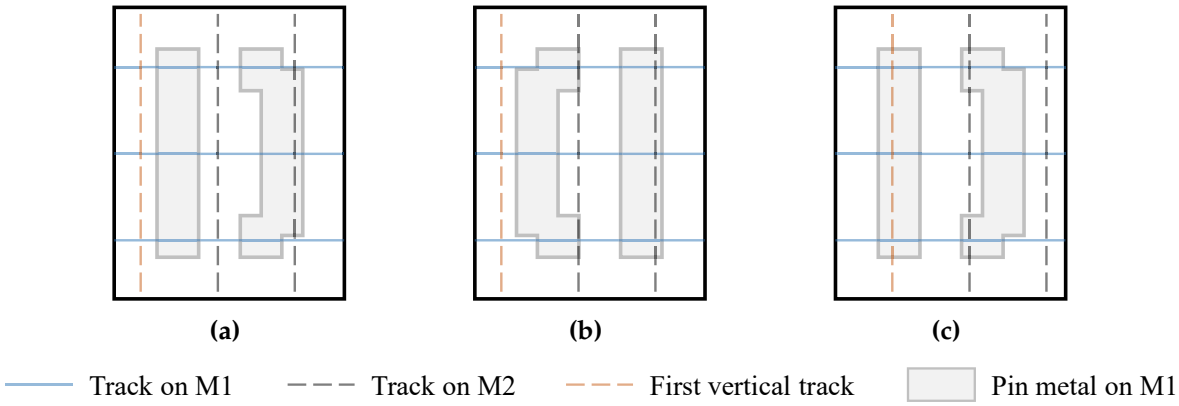


Figure 4.2: *Instance Patterns. (a) An instance of the same instance pattern as the rightmost instance in Figure 4.1a. (b) An instance of a different instance pattern due to different orientation. (c) An instance of a different instance pattern due to different track offset.*

width, resulting in a larger spacing requirement. Detecting such changes during the path searching stage can be challenging, emphasizing the crucial role of pin access analysis. Figure 4.3b depicts another PRL spacing violation between a routed via and the metal of the pin it intends to access. Moreover, Figure 4.3c demonstrates an EOL spacing violation between the metals of two routed vias on M2.

Almost all the design rules should be checked among maximal rectangles, which refers to the unique rectangles within a metal polygon that cannot be expanded further in width or height. For instance, the upper pin metal and the via metal in Figure 4.3a can be decomposed into three maximal rectangles as shown in Figure 4.3d. The blue rectangle represents an additional maximal rectangle that arises after accounting for the routed via, while the lower pin metal itself constitutes a maximal rectangle. In Figure 4.3d, the maximal rectangles outlined in black are referred to as *fixed rectangles*, originating solely from fixed metals. The maximal rectangle outlined in blue is referred to as a *routed rectangle* since it represents a newly formed maximal rectangle encompassing certain routed segments. The extraction of maximal rectangles from metal polygons is essential because the calculation of

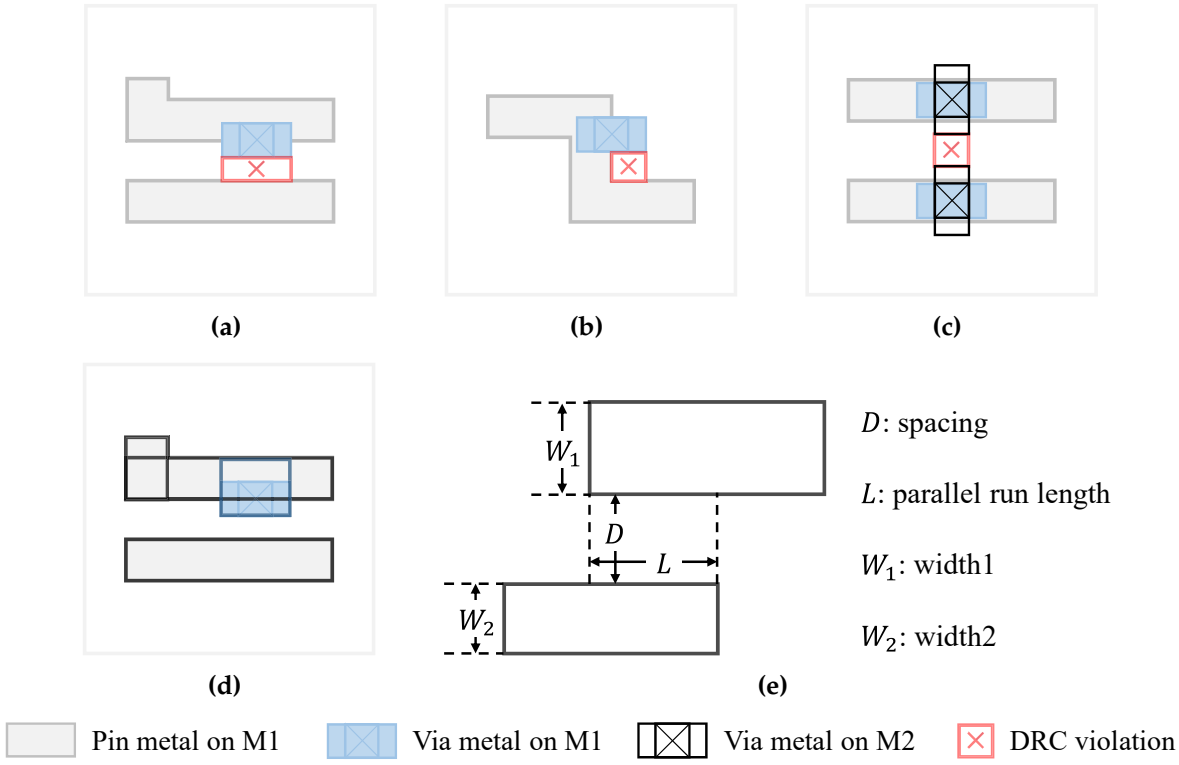


Figure 4.3: *Design Rule Checking. (a) A PRL spacing violation between a routed via's metal and the fixed metal of a different pin. (b) A PRL spacing violation between a routed via's metal and the metal of the pin it attempts to access. (c) DRC violations between two routed vias' metals. (d) The maximal rectangles decomposed from Figure 4.3a. (e) A PRL situation.*

spacing requirements for numerous design rules relies on the properties of these maximal rectangles, such as maximal width and parallel run length.

Algorithm 5 Design Rule Checking for Parallel Run Spacing

```
1: function ParaRunCheck(r, R)
2:     if the check is skippable then return false
3:     L ← parallel run length between r and R
4:     if L > 0 then
5:         W ← max(Width(r), Width(R))
6:         S ← SpacingTable(W, L)
7:         if Dist(r, R) < S then return true
       return false
```

Algorithm 5, for example, shows the algorithm for checking the parallel run spacing between a routed rectangle r and a fixed rectangle R. The parallel run spacing rule requires that two rectangles, as illustrated in Figure 4.3e, having a parallel run length L and maximum width $W = \max(W_1, W_2)$ should have a minimum spacing D specified by a lookup table ($D \geq$ SpacingTable(W, L)). To check the rule, we will first examine if the check is skippable (line 2), as there will not be a parallel run spacing violation if two rectangles overlap or there is no empty space between them at all. Next, if the parallel run length L is positive, we will calculate the maximum width W of the two rectangles and look up the spacing table to find the spacing S needed (line 3-6). The check will return *true* only if the spacing D between the two rectangles is smaller than S (line 7), indicating there is a parallel run spacing violation.

4.2 Methodology

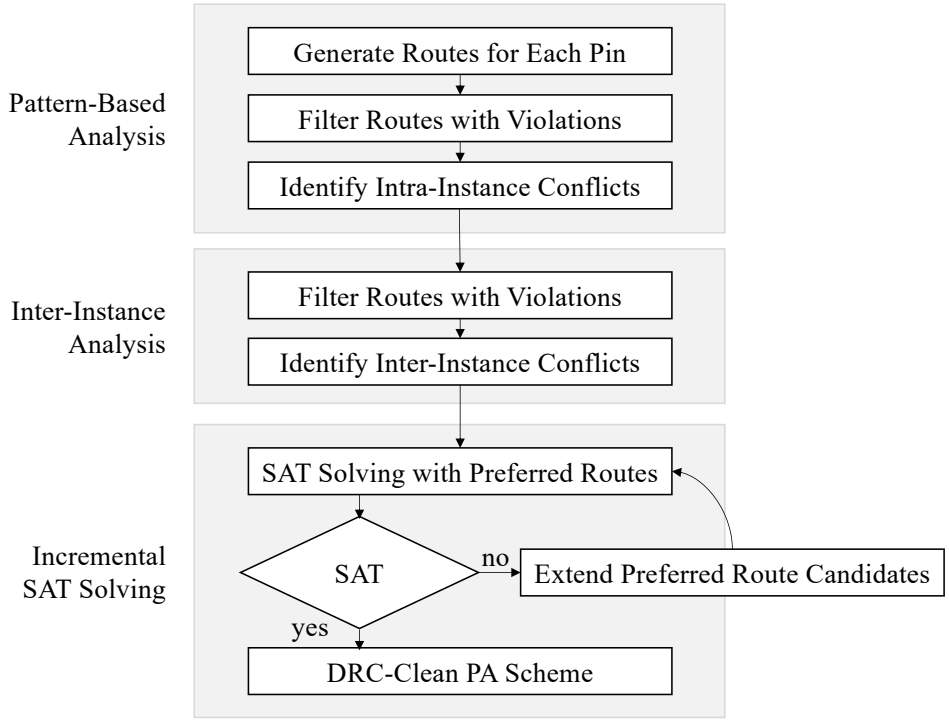


Figure 4.4: *Overall Flow.*

4.2.1 Overview of FastPass

The overall flow of FastPass is illustrated in Figure 4.4. To begin with, the pattern-based analysis is performed. Within each instance pattern, multiple candidate routes are generated for every pin. These routes serve the purpose of connecting each pin to a nearby grid point using a via and potentially a few wires. During the pattern-based analysis, we will conduct the first round of the candidate routes filtering, aiming to eliminate any route that would result in DRC violations with the fixed metals from the cell instance. After that, we identify the conflicts between routes associated with different pins within the same cell instance to form an *intra-instance conflict graph*.

In the next step, inter-instance analysis is carried out, where the candidate routes are filtered for the second time to remove those that will cause DRC violations

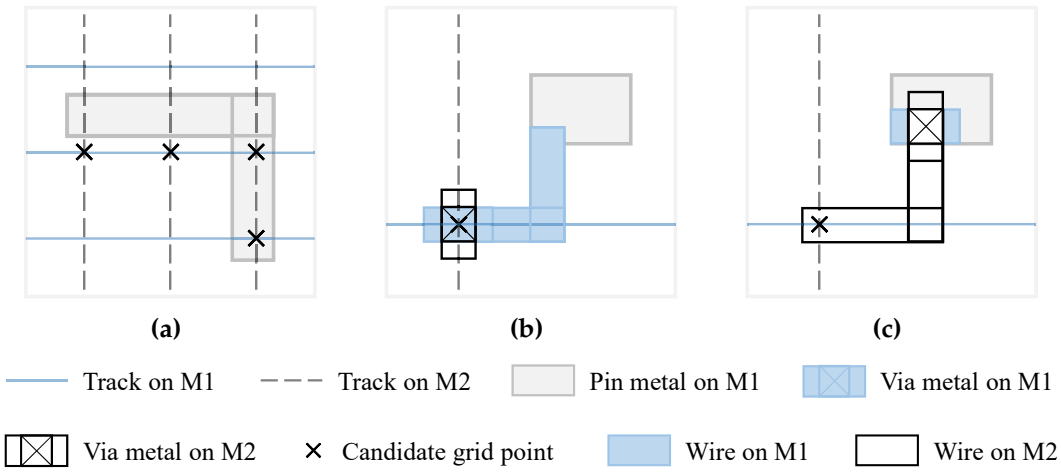


Figure 4.5: *Candidate Routes Generation. (a) Pin access grid points. (b) Pin access route with an on-grid via. (c) Pin access route with an off-grid via.*

with the fixed metals in neighboring cell instances. We then compute the conflicts between routes across different cells to generate an *inter-instance conflict graph*.

Finally, we conduct incremental SAT solving to find an optimized DRC-clean pin access scheme, where we solve SAT instances in an iterative manner. Each route will be assigned a cost based on its potential to complicate the detailed routing process. Initially, only the route with the lowest cost for each pin is enabled, prioritizing preferred routes. If the SAT instance is unsatisfiable with the available routes, the preferred candidate routes will be extended before the solver is invoked again. The procedure will repeat until a solution can be found.

4.2.2 Instance Pattern-Based Analysis

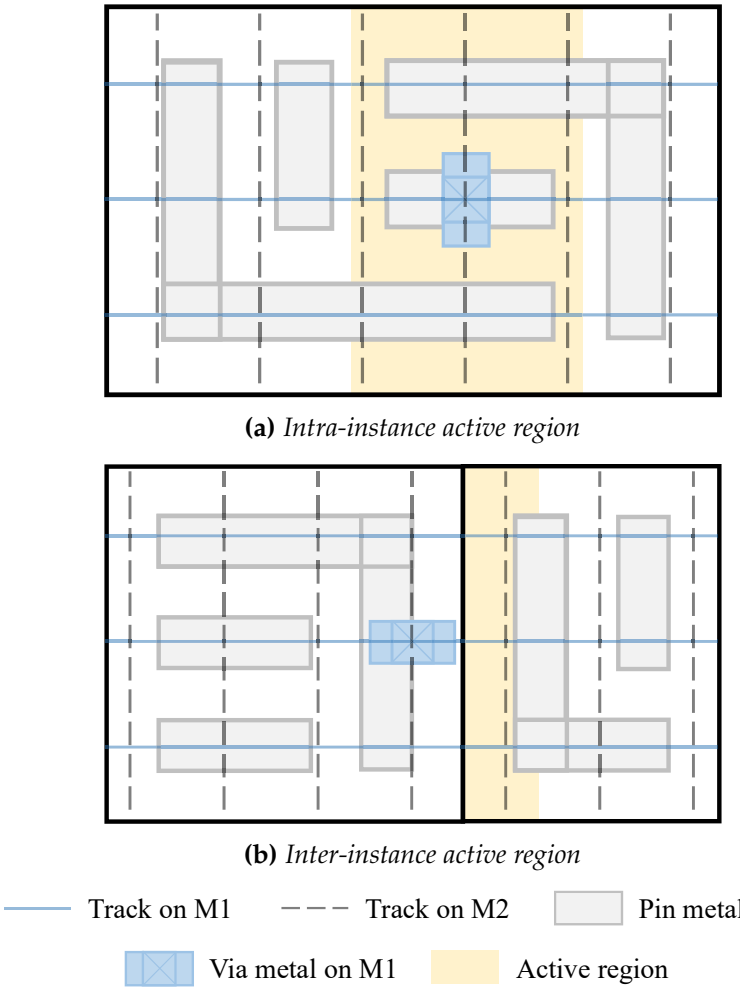
(a) Candidate Routes Generation: To analyze a unique instance pattern, we first identify a set of access grid points on M2 for each pin. Within each maximal rectangle of a pin shape, we incorporate all the grid points it includes into the set GP. In case a maximal rectangle does not contain any grid point, we incorporate

the nearest grid points into the rectangle in GP. For example, Figure 4.5a shows a pin shape with two maximal rectangles, and within these rectangles, a total of four grid points are identified.

Following that, we generate two types of routes for each grid point ($ap \in GP$) using different via types. Note that only via types with M2 metals in the same direction as the M2 tracks are considered. For the first type of routes, a via is used to connect a grid point on M2 to M1. Subsequently, two short wire segments on M1 are used to establish a connection from the center of the via to the nearest pin metal, as depicted in Figure 4.5b. If the via center is already within the pin metal, the wire segments can be saved. For the second type of routes, wires on M2 are used to route from the grid point to a position directly above the pin metal, where a via can be placed with its M1 metal fully contained within the pin metal, as illustrated in Figure 4.5c.

Typically, the first type of routes is favored since it consumes less routing resources on M2. However, there are cases where certain pins cannot be accessed using the first type, particularly when the layout on M1 is too compact.

Once the candidate routes are generated, all the routes within an instance pattern will be examined together to identify and filter out those causing DRC violations with the fixed metals. To conduct this violation checking, it is necessary to decompose all fixed metals and routed metals into maximal rectangles as discussed in Section 4.1.3. In the rest of this chapter, we use "rectangle" to refer to maximal rectangle for brevity. Typically, the checking of DRC violations related to a rectangle involves finding all other rectangles within a specified distance using R-tree [43], which is an efficient data structure for spatial indexing. Nonetheless, as the number of rectangles increases and similar queries are performed repeatedly, the R-tree approach can become slow. Therefore, we adopt a completely different approach



(a) *Intra-instance active region*

(b) *Inter-instance active region*

Figure 4.6: *Intra-Instance and Inter-Instance Design Rule Checking. A routed rectangle will cause violations with only a few fixed rectangles within the active region.*

Algorithm 6 Intra-Instance DRC Violation Check

1: Sort all routed rectangles in non-decreasing order of x: r_1, r_2, ... r_n.
2: Sort all fixed rectangles in non-decreasing order of x: R_1, R_2, ... R_N.
3: $A \leftarrow \emptyset$, $j \leftarrow 1$
4: **for** $i = 1, 2, ..., n$ **do**
5: Shift the active region for routed rectangle r_i
6: **while** R_j has overlap with the active region **do**
7: $A \leftarrow A + \{R_j\}$
8: $j \leftarrow j + 1$
9: **for** each fixed rectangle $R' \in A$ **do**
10: **if** R' falls to the left of the active region **then**
11: $A \leftarrow A - \{R'\}$
12: **else**
13: Check violations between r_i and R'

for efficient design rule checking.

Our algorithm takes advantage of the fact that cell instances have a limited height. As a result, a routed rectangle may cause violations with only a few fixed rectangles within a small distance in the x direction (horizontal), represented by the yellow region in Figure 4.6a. We refer to this region as the *active region* of a rectangle. The width of an active region can be determined based on the width of the rectangle and the maximum spacing requirement.

With this fact, we check the DRC violations for all routes together by a sweep-line algorithm, which can be described as in Algorithm 6. We first sort the routed rectangles and fixed rectangles in non-decreasing order of their x coordinates respectively (line 1-2). We will then initialize the active set, i.e. the set of fixed rectangles within the active region, as an empty set ($A \leftarrow \emptyset$). The active region is initially outside the left boundary of the instance at the beginning. As we iterate through each routed rectangle from left to right, we proceed to adjust the the active region by shifting the left and right boundaries (line 4-5). The active set is updated by including the fixed rectangles that enter from the right side (line 6-8)

and removing those that exit from the left side of the active region (line 9-11). For each routed rectangle, we will only check it with the fixed rectangles within the corresponding active region (line 9-13). The routes having a rectangle with DRC violations will be filtered.

Let N denote the number of fixed rectangles, n represent the number of routed rectangles, and m signify the maximum number of fixed rectangles within the active region. Excluding the sorting step, the time complexity of the aforementioned algorithm will be $O(N + nm)$. This complexity arises from the requirement of $O(N)$ time for including and removing each fixed rectangle from the active set, and an additional $O(nm)$ time for checking violations between each routed rectangle and the fixed rectangles in the corresponding active set. It is worth noting that in a single cell instance, the values of N, n, and m are typically small. Consequently, Algorithm 6 can be highly efficient in practice.

(b) Intra-Instance Conflict Graph: Once the routes that violate DRC rules with fixed metals have been filtered out, the conflicts (violations) between routes for different pins within the same instance will be examined. This step is crucial for constructing an intra-instance conflict graph. Similar to the violation checking with fixed metals, a routed rectangle only requires examination against routed rectangles of other pins within the active region. The checking algorithm closely resembles Algorithm 6, with the distinction that the active set is utilized to keep track of other routed rectangles within the active region instead of fixed rectangles. Upon completion of the checking process, an intra-instance conflict graph is formed, where each route is represented as a vertex. An edge is established between two vertices (routes) if they are from different pins and cannot be selected simultaneously.

4.2.3 Inter-Instance Analysis and Complete Conflict Graph

After completing the instance pattern-based analysis, we obtain a set of routes and an intra-instance conflict graph for each cell instance. These routes are compatible with the fixed metals within their respective cell instances, and the conflict graph indicates the compatibility between routes within the same instance. However, routes located near the left or right boundary of an instance may still have the potential to conflict with fixed metals or other routes in neighboring instances, as illustrated in Figure 4.6b. Fortunately, the same method employed for checking intra-instance violations can also be applied in this context. Routes located near the boundary of an instance only need to be assessed against the fixed metals and routes within a small active region (highlighted in yellow in Figure 4.6b) near the boundary of the neighboring instance. Following this process, we can efficiently filter out routes that are incompatible with the fixed metals in the neighboring instances and further identify conflicts between routes from different instances. Those conflicts together with the intra-instance conflict graph will then be used to construct the inter-instance conflict graph.

A toy example is presented in Figure 4.7 for clearer demonstration. Note that the conflict graph involving real standard cells can be more complex than the one depicted in this example. Figure 4.7a showcases three cell instances of the same instance pattern in the same placement row. Following the instance pattern-based analysis, six routes are generated, each featuring a single via as illustrated in Figure 4.7b. Additionally, an intra-instance conflict graph is established as displayed in Figure 4.7d. Notably, $t_{X,i}$ denotes the i^{th} route for pin X. Upon conducting the inter-instance analysis and assuming no existing routes are filtered out, the complete conflict graph is obtained, as depicted Figure 4.7c. The inter-instance edges are

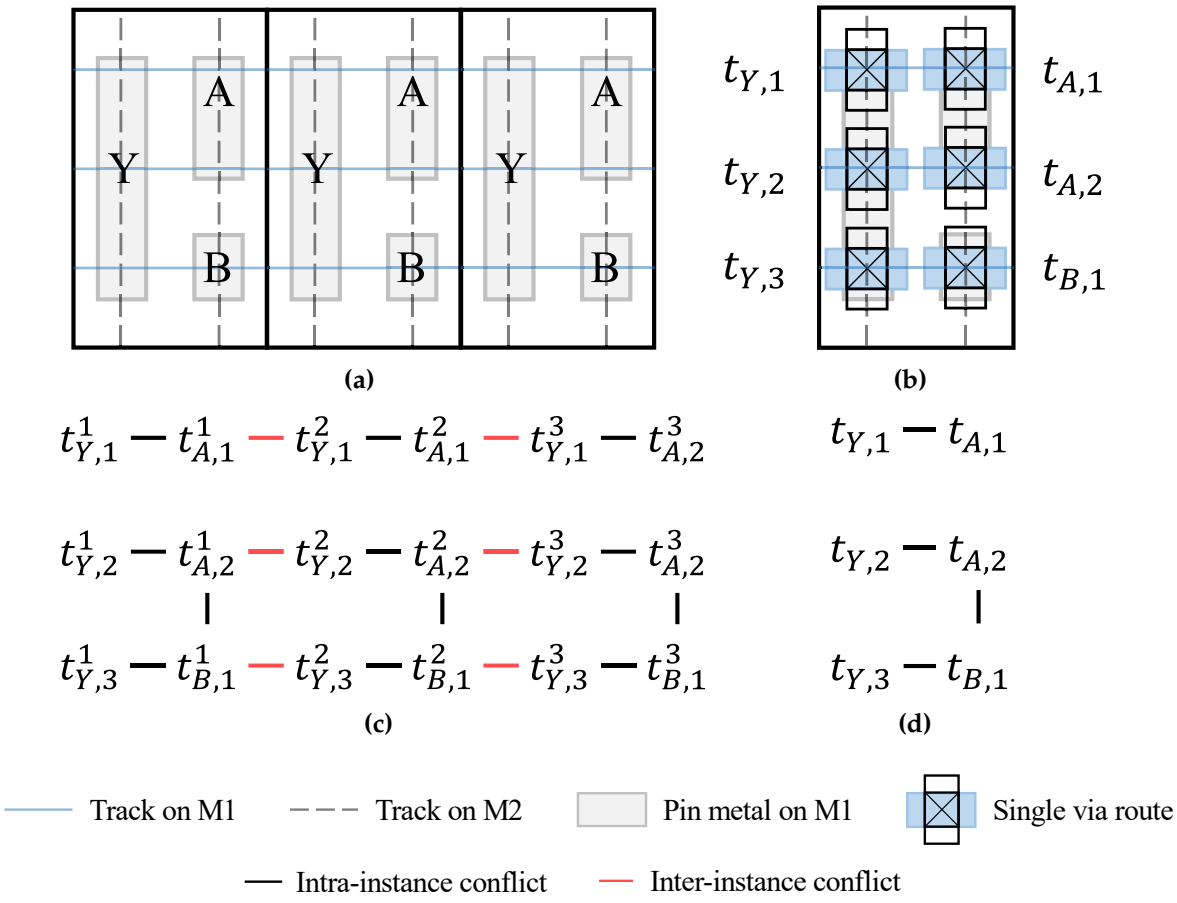


Figure 4.7: *Conflict Graph. (a) Three identical cell instances in a row. (b) Possible routes for the pins in the instance. (c) Complete conflict graph. (d) Intra-instance conflict graph.*

highlighted in red, and $t^j_{X,i}$ denotes the i^{th} route for pin X of the j^{th} instance.

4.2.4 Route Selection by Incremental SAT Solving

With DRC-clean candidate routes and the complete conflict graph, we introduce an incremental SAT-based approach for route selection, such that each net pin is assigned one route to form an optimal or close to optimal pin access scheme without any inter-route conflict.

(a) Division into Independent Subproblems: It is observed that the route selection problem for the entire design can be broken down into multiple independent subproblems. Each subproblem focuses on a small group of pins and can be resolved quickly. Depth-first search (DFS) is performed on the conflict graph to identify all the connected components, and each component will be treated as an individual problem instance. During DFS, to prevent scenarios where a pin's candidate routes present in multiple subproblems, we introduce edges between routes from the same pin. Therefore, the initially large problem can be efficiently resolved. In the rest of this section, we focus on one subproblem for clarity.

(b) Problem Definition with Notations: Consider the set of pins P, we denote T_i as the set of candidate routes for pin p_i and let $t_{i,j} \in T_i$ be the j^{th} candidate route of pin p_i. With the conflict graph $G = (V, E)$, a bijection (one-to-one mapping) exists between the set of routes and the set of vertices V. With an abuse of notation, for a route $t_{i,j}$, we denote its corresponding vertex in V as $t_{i,j}$, too.

To indicate our preference on different routes, each route $t_{i,j}$ is assigned a cost tuple $cost(t_{i,j}) = \langle c_1, c_2 \rangle$, which will be calculated as follows.

$$c_1 = \begin{cases} 1, & \text{if } ap(t_{i,j}) \text{ is out-of-guide,} \\ 0, & \text{otherwise,} \end{cases} \tag{4.1}$$

$$c_2 = L1Dist\big(ap(t_{i,j}), net(p_i).center\big), \tag{4.2}$$

where $ap(t_{i,j})$ means the access grid point of route $t_{i,j}$ and $net(p_i)$ represents the net to which pin p_i belongs. Note that the global route guides are a set of regions where detailed routing is preferred. Hence, by introducing c_1, we first consider routes that has grid points within the guides. Additionally, the term c_2 allows us to prioritize routes that are closer to the bounding box center of the net, as this can potentially

result in a shorter total half-perimeter wire length (HPWL). Note that the choice of cost function can be flexible. Other tailor-made functions for better routability can also be used, for example, history cost.

We assume that in each T_i, routes have been sorted in lexicographic order (e.g., $\langle 0, 500\rangle < \langle 0, 1000\rangle < \langle 1, 200\rangle$). In the route selection problem, we want to select a set of routes AS, such that

(1) for each pin $p_i \in P$, there must be a route $t_{i,k} \in AS$;

(2) for all $t_{i_1,j_1} \neq t_{i_2,j_2} \in AS$, we require that the two routes do not conflict with each other, i.e., $(t_{i_1,j_1}, t_{i_2,j_2}) \notin E$;

(3) we want to use preferred routes (i.e., the first few candidates) as much as possible.

(c) Boolean Formulation for Route Selection: For this problem, we propose a neat Boolean formula $\mathcal{F}$ with two types of constraints,

$$\mathcal{F} \equiv \mathcal{P} \wedge \mathcal{C}. \tag{4.3}$$

In the given equation, $\mathcal{P}$ ensures that at least one route is selected for each pin. $\mathcal{C}$ incorporates information from the conflict graph, indicating that two conflicting routes cannot be chosen simultaneously. With a set of Boolean variables S, where $s_{i,j} \in S$ is asserted true if route $t_{i,j}$ is selected, the constraints $\mathcal{P}$ and $\mathcal{C}$ can be expressed as follows.

$$\mathcal{P} \equiv \bigwedge_{p_i \in P} \bigvee_{t_{i,j} \in T_i} s_{i,j}, \tag{4.4}$$

$$\mathcal{C} \equiv \bigwedge_{(t_{i_1,j_1}, t_{i_2,j_2}) \in E} (\neg\, s_{i_1,j_1} \vee \neg\, s_{i_2,j_2}). \tag{4.5}$$

By feeding $\mathcal{F}$ into an SAT solver, we can readily obtain a solution that represents a DRC-clean pin access scheme if one exists. Furthermore, if multiple access schemes are required to enhance detailed routability, we can easily introduce the negation of the current variable assignment as a new clause to $\mathcal{F}$ and perform SAT solving once more to obtain a distinct solution. Nonetheless, since an SAT solver does not incorporate any optimization objectives, the resulting pin access scheme may include numerous access grid points that are located outside the guides or are relatively distant from the net centers. Such outcomes may not be favorable for detailed routing. To this end, we introduce an incremental SAT solving approach that enables the generation of an optimal or nearly optimal solution.

(d) Incremental SAT Solving: The concept of *incremental SAT solving with assumptions* is first introduced in Minisat [34]. It allows for the inclusion of a set of assumptions, expressed as unit clauses, during a single invocation of the SAT solver. This feature empowers users to (i) enable or disable specific clauses, and (ii) pre-assign certain variables as needed [91]. As a result, incremental SAT solving can be more powerful compared to conventional SAT solving.

When the solver is invoked with assumptions, it will either (i) reports satisfiable (SAT) and produces a solution that fulfills all the assumptions or (ii) reports unsatisfiable (UNSAT).

In the case of unsatisfiability, the solver will return a subset of assumptions that are sufficient to make the current invocation UNSAT [38]. This subset can be utilized to construct new assumptions for subsequent incremental runs.

(e) Incremental SAT Flow for Route Selection: Consider Equation (4.3), the SAT solver has a good chance of picking less preferred routes because such assignments

exist in the search space. With assumptions, we can manually adjust the search space of an SAT problem instance. For example, by having the assumption $\neg s_{i,j}$, we can eliminate feasible solutions where $s_{i,j}$ is assigned 1. The usage of route $t_{i,j}$ is thus disabled.

As shown in Figure 4.4, we propose the following incremental SAT solving approach, which can be considered as an iterative process in which we dynamically adjust the search space of an SAT problem. The detailed incremental SAT flow is described in Algorithm 7.

(1) We start the SAT solving with only the best route enabled for each pin (line 5-6). To achieve this, we use assumptions to disable other routes (line 8-12).

(2) If the solver reports SAT, we can stop the process and convert the assignment to an optimized pin access scheme (line 13-15).

(3) Otherwise, there must be some insolvable conflicts between the routes in the current search space. Therefore, we will find a set of pins P' that need to be extended based on the returned assumptions (line 20-21) and enable one more candidate route for each pin in P' (line 22-23). In this way, we increase the chance to get a feasible solution in the next incremental run while introducing as few suboptimal routes as possible.

Note that if the solver reports UNSAT when no assumptions are passed to the solver (i.e., no routes are disabled), we can conclude that no feasible solutions exist for the route selection problem (line 18).

In contrast to normal SAT solving, the proposed incremental flow can involve multiple incremental runs of an SAT solver, with each run encompassing a relatively small search space. Hence, in practice, invoking the SAT solver multiple times within the proposed incremental flow will not result in longer runtime. More

Algorithm 7 Incremental SAT-Based Route Selection

Initialize the SAT solver $solver$
for clause $c \in \mathcal{F}$ **do**
    $solver.addClause(c)$
$L \leftarrow \{\}$ ▷ Stores the index of the last enabled route for each pin.
for $i = 1, 2, ..., |P|$ **do**
    $L[i] \leftarrow 1$ ▷ Only enable the most preferred routes at the beginning.
while $true$ **do**
    Create the set of assumptions $\mathcal{S} \leftarrow \emptyset$
    for $i = 1, 2, ..., |P|$ **do**
        for $j = L[i] + 1, L[i] + 2, ..., |T_i|$ **do**
            $\mathcal{S}.add(\neg\, s_{i,j})$
    $isSAT = solver.solve(\mathcal{S})$
    if $isSAT = true$ **then**
        $AS \leftarrow convertToRoutes(solver.assignment)$
        return AS ▷ The pin access scheme.
    else
        if $S = \emptyset$ **then**
            Report UNSAT and **return**
        $P' \leftarrow \{\}$ ▷ The set of pins to relax.
        for each literal $l \in solver.conflicts$ **do**
            $P'.add(getPin(l))$
        for $p_i \in P'$ **do**
            $L[i] \leftarrow min(L[i] + 1, |T_i|)$

importantly, by progressively expanding the search space, we can ultimately achieve an optimal or near optimal solution using just an SAT solver, instead of resorting to potentially more time-consuming approaches such as MAX-SAT or integer linear programming (ILP).

4.2.5 Integration into the Detailed Router

We further integrate FastPass into Dr. CU [73, 21] to validate the effectiveness of our proposed approach. However, with FastPass integrated, we observe an increasing number of min-area rule violations[1] and spacing rule violations in the detailed routing solution for some of the designs. Next, we will discuss the reasons and how we make FastPass more compatible with Dr. CU.

(a) Accurate Min-Area Patching: Compared with the original pin access strategy of Dr. CU, FastPass introduces more vias to M2 layer, which will increase the demand for min-area patches. However, Dr. CU's min-area patching algorithm is grid-based. It will either patch more than needed or give up fixing the min-area violation when it is doable. Figure 4.8a shows an example of this problem, where Dr. CU fails to patch when there is enough space to satisfy the min-area requirement. To handle such cases, we add an accurate min-area patching step at the end of the routing flow, which does not rely on the grid graph structure. We will first shrink the over-patched wires to create more space. After that, for each min-area rule violation, we calculate the minimum amount of patch that we need to satisfy the requirement. As shown in Figure 4.8b, when adding the patch, we will explore one end of the shape first (along the preferred routing direction), being aware of possible spacing

[1]The min-area rule specifies the minimum area required for polygons on each layer. It mandates that all polygons should have an area greater than or equal to the specified area value [87].

violations with nearby objects. If the total area is still too small, we will continue extending the patch on the opposite side.

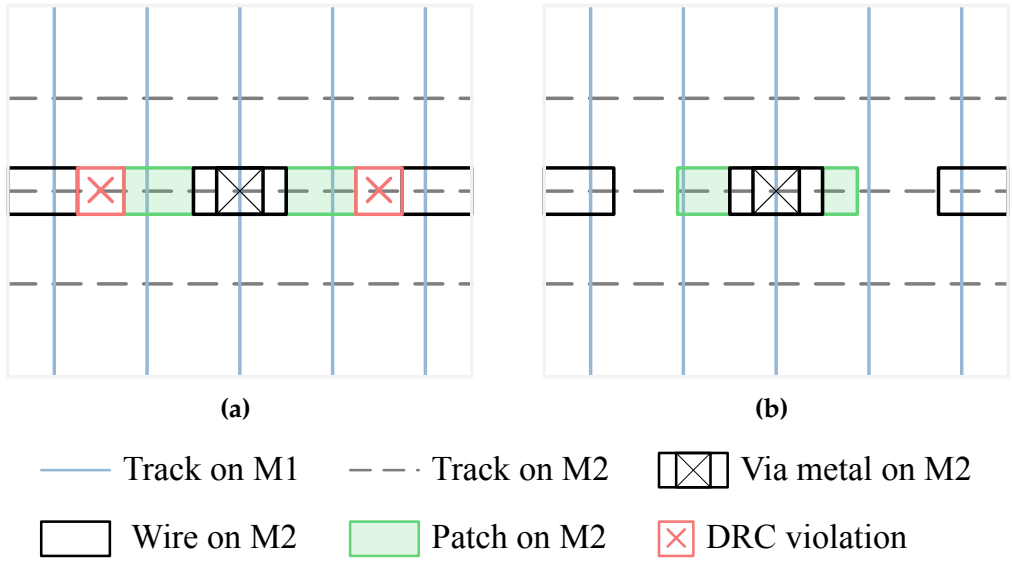


Figure 4.8: *Min-Area Patching. (a) Dr. CU fails to fix the min-area rule violation with the grid-based patching algorithm. (b) Accurate min-area patching.*

(b) Improved History Cost: Given the increased number of vias on M2 layer, the probability of having spacing violations around via metals rises. During rip-up and reroute, Dr. CU identifies both short and spacing violations to determine which nets will be rerouted. But it will only mark history cost for short segments as Figure 4.9a shows. As a result, most of the spacing violations will still remain in the end. To address this issue, we revise the history cost scheme in Dr. CU such that it will also mark routing edges that are close to the spacing violation as illustrated in Figure 4.9b.

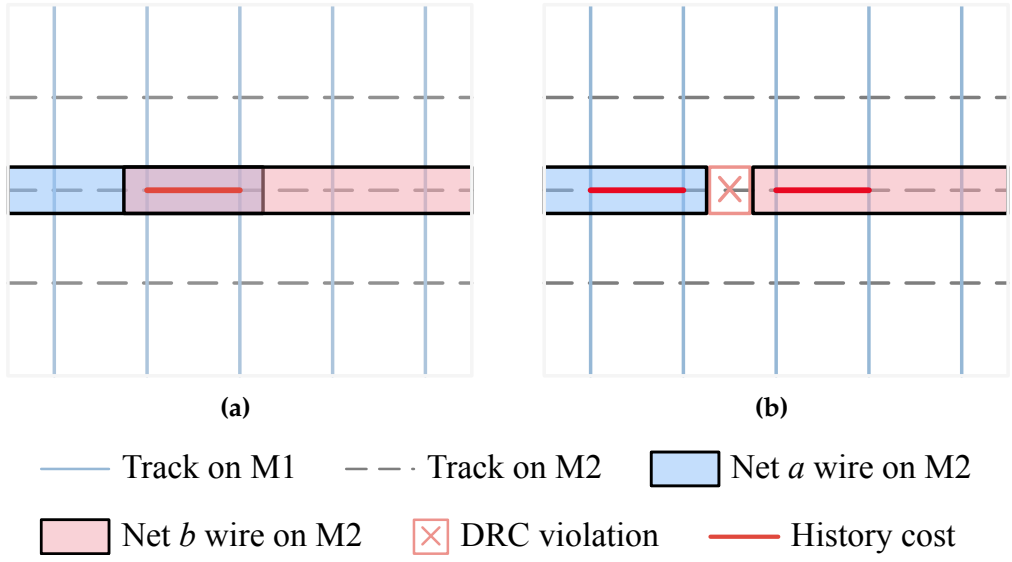


Figure 4.9: *History cost for (a) short violations and (b) spacing violations.*

4.3 Experimental Results

We implement FastPass in C++, and MiniSat 2.2 [34] is used for SAT solving. The design rules are checked by Innovus 20.14 [14] with the evaluation scripts from the ISPD 2018 Initial Detailed Routing Contest [87]. All the experiments are conducted on a Linux server with a 2.90 GHz Intel Xeon CPU. We conduct a series of experiments on the ISPD 2018 benchmark suite [87], which contains cases derived from real industrial designs. Detailed statistics of the benchmarks are listed in Table 4.1. In Section 4.3.1 and Section 4.3.2, we compare the runtime of FastPass with the state-of-the-art PAAF [60]. In Section 4.3.3, we study the performance of the proposed incremental SAT solving technique. In Section 4.3.4, we show that the proposed methodology can be extended for advanced technology nodes. We further integrate FastPass into Dr. CU and study how it will affect the detailed routing quality in Section 4.3.5.

Table 4.1: *Benchmark Statistics*

Benchmark	Tech (nm)	# Cell Instances	# Macros	# Nets	# IO	# Layers
ispd18_test1	45	8,879	0	3,153	0	9
ispd18_test2	45	35,913	0	36,834	1,211	9
ispd18_test3	45	35,973	4	36,700	1,211	9
ispd18_test4	32	72,094	0	72,401	1,211	9
ispd18_test5	32	71,954	0	72,394	1,211	9
ispd18_test6	32	107,919	0	107,701	1,211	9
ispd18_test7	32	179,865	16	179,863	1,211	9
ispd18_test8	32	191,987	16	179,863	1,211	9
ispd18_test9	32	192,911	0	178,857	1,211	9
ispd18_test10	32	290,386	0	182,000	1,211	9

Table 4.2: *Comparison between FastPass and the state-of-the-art pin access analysis framework [60] on the ISPD 2018 Benchmark [87]*

Benchmark	# Cell Instances	# Instance Patterns	# Pins	# Candidate Routes	# Conflict Edges	TOP Time (s)	FastPass Time (s)	Speedup
ispd18_test1	8,879	182	17,203	90,601	4,928	6.3	**0.2**	39.3×
ispd18_test2	35,913	222	159,201	785,600	44,399	10.5	**0.9**	11.8×
ispd18_test3	35,973	227	159,703	788,866	44,348	13.1	**0.9**	14.9×
ispd18_test4	72,094	2,725	318,245	4,242,600	106,053	118.5	**5.3**	22.3×
ispd18_test5	71,954	2,733	318,195	3,455,946	7,554,606	124.6	**7.2**	17.4×
ispd18_test6	107,919	2,886	475,541	5,125,632	11,072,916	143.5	**10.1**	14.2×
ispd18_test7	179,865	148	793,289	8,405,001	16,884,133	94.7	**11.6**	8.2×
ispd18_test8	191,987	150	793,289	8,405,271	16,902,913	114.9	**11.7**	9.8×
ispd18_test9	192,911	136	791,761	8,405,332	16,907,027	72.9	**11.7**	6.2×
ispd18_test10	290,386	144	811,761	8,614,315	17,393,590	105.9	**12.5**	8.5×
Avg.	-	-	-	-	-	80.5	**7.2**	15.3×

4.3.1 Comparison with TOP

We compare our framework with the known best PAAF - TOP[2] [60]. As both frameworks can be easily parallelized, a single thread is used in this experiment for a fair comparison. Quantitative results are shown on the right-hand side of Table 4.2. The same table also presents statistics related to the problem scale and the conflict graph constructed by FastPass. For most designs, FastPass generates approximately ten candidate routes on average per pin, and the number of identified conflict edges is roughly double the count of candidate routes. Both TOP and FastPass can achieve DRC-clean pin access schemes with no failed pins. Furthermore, it is worth noting that FastPass is on average 15.3$\times$ faster than TOP, demonstrating its strong potential in scenarios where placement changes frequently and pin access analysis needs to be performed repeatedly.

4.3.2 Runtime Decomposition

Both the runtime of TOP and FastPass can be roughly divided into two parts: preparation time and solving time. For preparation, the unique instance patterns are analyzed, and DRC-clean access points and routes are generated. For solving, a DRC-clean pin access scheme is generated by DP in TOP and by incremental SAT in FastPass. The actual runtime decomposition of the two algorithms is shown in Figure 4.10. We observe that our preparation with sweep-line design rule checking is much faster than TOP for most cases except for *ispd18_test9* and *ispd18_test10*. While SAT-based methods are commonly regarded as slower, our solving part is shown to be exceptionally fast due to the inherent small size of the majority of independent subproblems. Furthermore, as our pin access scheme is constructed

[2]The source code is available at https://github.com/ABKGroup/TritonRoute-WXL.

to be correct, there is no need for additional DRC checking when compared to the DP-based algorithm in TOP. For the last two benchmarks, TOP uses less preparation time for the following reasons.

(1) FastPass checks inter-instance violations during preparation while TOP leaves them to the solving stage.

(2) In the last two cases, the cell instances are of relatively simple structures but compactly placed, which makes those inter-instance checks more time-consuming.

This also explains why TOP spends much more time on the solving stage for these two cases.

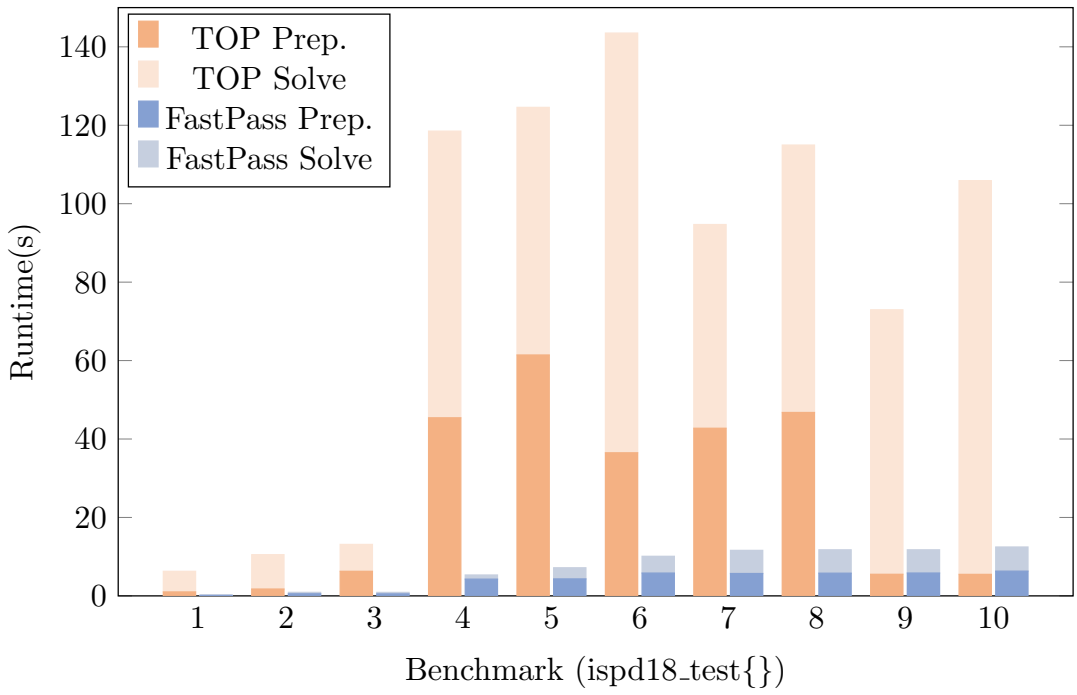


Figure 4.10: *Runtime decomposition of TOP and FastPass.*

4.3.3 Effectiveness of Incremental SAT Solving

As discussed in Section 4.2.4, rather than simply selecting a valid route for each pin, we utilize incremental SAT solving to enhance the quality of the pin access scheme. In this work, two metrics are employed to assess the quality of a pin access scheme: the number of out-of-guide access grid points (OFG) and the total half-perimeter wire length (HPWL) of the nets. Given that detailed routers typically prioritize path searching within the route guides, an excessive number of OFGs can result in prolonged path searching and degradation of routing quality. Additionally, the total HPWL can serve as a lower bound of the actual routed wire length, which makes it a potential target for optimization.

Table 4.3: *Route Selection Results with/without Incremental SAT Solving*

Benchmark	TOP		SAT-Unsorted		SAT-Sorted		SAT-Incremental	
	# OFG	HPWL	# OFG	HPWL	# OFG	HPWL	# OFG	HPWL
ispd18_test1	860	62,754	860	62,772	33	61,532	8	61,507
ispd18_test2	7,188	1,338,360	8,039	1,338,250	1,010	1,322,540	688	1,321,953
ispd18_test3	6,992	1,457,780	7,158	1,457,680	1,595	1,442,440	1,092	1,441,970
ispd18_test4	50,207	2,158,830	37,586	2,158,290	2,137	2,123,870	1,815	2,123,423
ispd18_test5	46,445	2,164,850	45,876	2,164,150	3,457	2,132,330	2,161	2,131,273
ispd18_test6	73,750	2,942,230	73,322	2,941,600	5,422	2,894,670	3,483	2,893,068
ispd18_test7	107,177	5,054,720	96,965	5,053,730	2,828	4,970,390	1,466	4,968,416
ispd18_test8	108,038	5,076,690	97,926	5,075,710	2,904	4,992,430	1,464	4,990,500
ispd18_test9	130,448	4,495,240	121,723	4,494,250	4,088	4,411,420	2,637	4,409,575
ispd18_test10	136,333	5,621,630	126,904	5,620,730	4,433	5,537,060	2,785	5,535,057
Geo. Mean Ratio	32.158	1.016	30.554	1.016	1.733	1.000	1.000	1.000

* OFG: Out-of-guide access grid points. The unit for HPWL is μm. Geo. Mean: Geometric mean.

In Table 4.3, we compare the quality of the pin access scheme found by TOP and FastPass. For FastPass, we compare three different strategies, namely SAT-Unsorte, SAT-Sorted, and SAT-Incremental. For SAT-Unsorted, we will feed the candidate routes into the SAT solver without any prior sorting. With this approach, the quality

of pin access schemes produced by FastPass is similar to that of TOP in terms of both the number of OFGs and the total HPWL. For SAT-Sorted, we will sort the candidate routes in ascending order based on their associated costs as discussed in Section 4.2.4 before SAT solving. Since the solver tends to select the first option presented to it, sorting beforehand helps to reduce the number of OFGs by 94.3% and reduce the total HPWL by 1.5% compared with SAT-Unsorted. Lastly, we employ incremental SAT for route selection, which further reduces the number of OFGs by 42.3% compared to the SAT-Sorted approach.

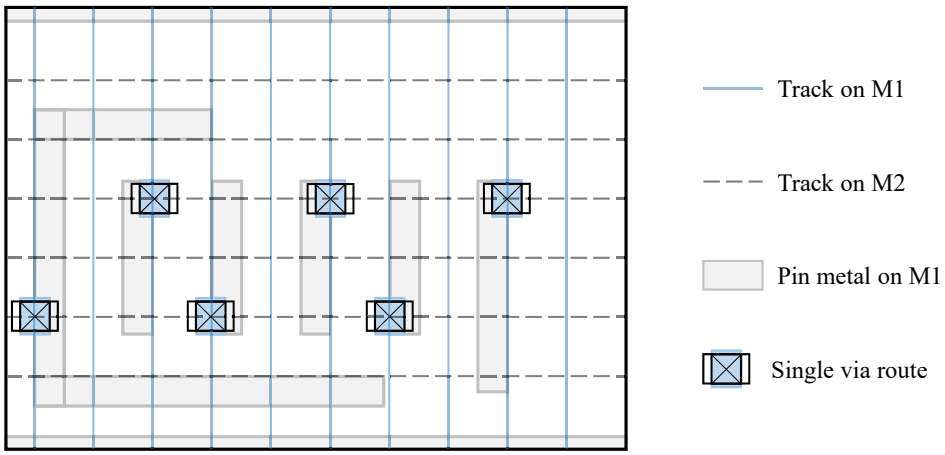


Figure 4.11: *Pin access result for an AOI221 cell in mor1kx [90].*

4.3.4 Supporting Advanced Technology Nodes

To show that our proposed methods can be applied to advanced technology nodes as well, we synthesize an open-source processor IP core *mor1kx* [90] (80K instances, 515 instance patterns) with the ASAP 7nm library [26]. We observe that FastPass can generate DRC-clean pin access result in 4.1 seconds with simple adaptation[3].

[3]Necessary modifications are made to FastPass to deal with different design rules in the advanced technology node. For example, we add the design rule check for the end-of-line keepout spacing rule and rule out all routes with off-grid routing segments.

Figure 4.11 shows the pin access scheme for an AOI221 cell from *mor1kx*.

4.3.5 Integration into Dr. CU

We further integrate FastPass into Dr. CU [73, 21]. For brevity, we refer to "Dr. CU with FastPass integration" as "FastPass" in the rest of this section. The detailed routing results are shown Table 4.4, where we compare the performance of Dr. CU and FastPass. The best recorded total DRC number for each test case is highlighted in bold. We set the maximum rip-up and reroute (RRR) iteration number to be eight and run each test case with sixteen threads.

The upper part of Table 4.4 shows the results on ISPD 2018 benchmarks with the contest global routing result, that is, the route guides. It is observed that with FastPass integrated, Dr. CU is able to produce much better results with 87.5% less short area and 72.4% fewer DRC violations.

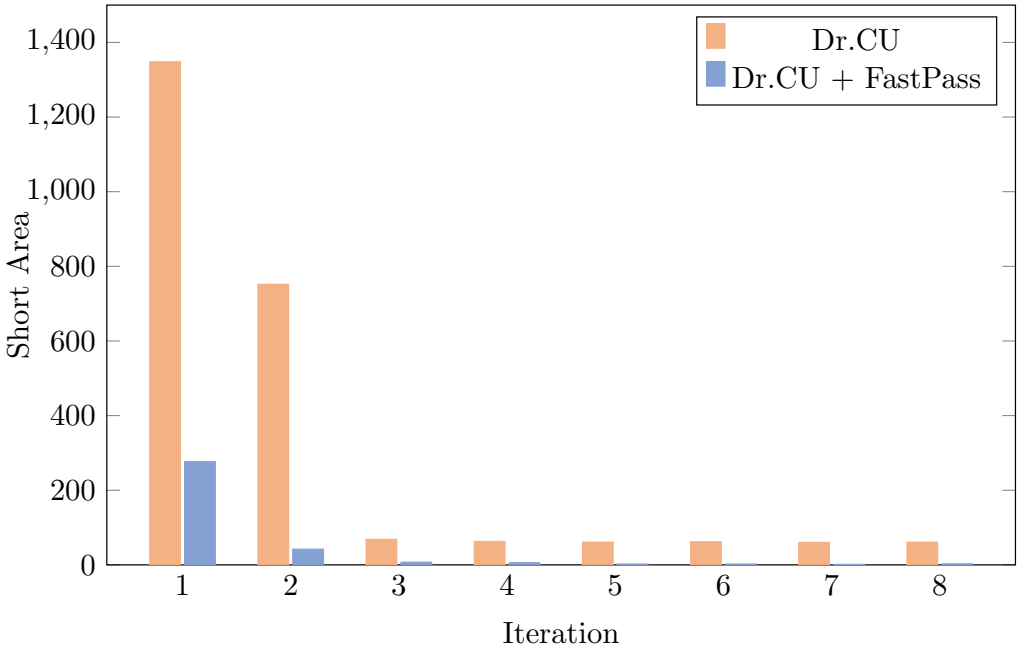
Figure 4.12a and Figure 4.12b depict the short area and total DRC violation numbers obtained for different maximum RRR iteration numbers for *ispd*18_*test*5. The integration of FastPass brings substantial improvements in detailed routing quality, particularly during the early routing stages (e.g., when the iteration number is one or two). Moreover, FastPass maintains its advantage throughout the process as the iteration number increases. Notably, with larger iteration numbers, there is little short area in the detailed routing results produced by FastPass.

In Figure 4.13, we compare the pin access results between Dr. CU and FastPass. When a routed wire and pin shape overlap, maximal rectangles with larger width can appear, leading to larger spacing requirements. Capturing those changes can be difficult during path search. As shown in Figures 4.13a and 4.13c, without proper pin access analysis, it can be difficult for Dr. CU to capture spacing violations where complicated pin shapes are involved. Meanwhile, FastPass is able to provide the

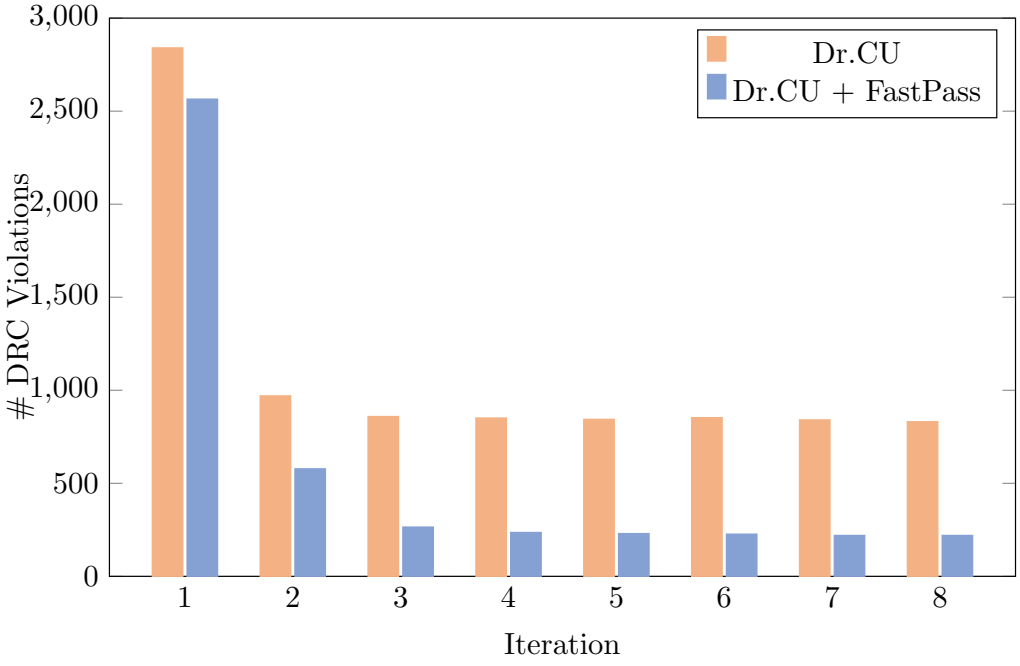
Table 4.4: *Detailed routing results with route guides from the ISPD2018 Benchmark [87] and CUGR [79]*

		Benchmark	Wire Length	# Via	# Short	Short Area	# MAR	# Spacing	Total DRC #	Time (s)
ISPD 2018 Guides	Dr. CU	ispd18_test1	433,572	32,391	1	0.1	0	2	3	20
		ispd18_test2	7,832,401	325,574	0	0.0	0	57	57	238
		ispd18_test3	8,719,013	318,401	159	220.5	0	89	248	1,024
		ispd18_test4	26,400,469	725,969	157	30.0	96	583	836	3,342
		ispd18_test5	27,800,625	965,147	335	60.7	143	355	833	2,170
		ispd18_test6	35,701,358	1,480,108	24	9.7	288	546	858	985
		ispd18_test7	65,168,700	2,401,573	618	146.0	340	185	1,143	4,381
		ispd18_test8	65,464,844	2,411,369	612	142.8	393	175	1,180	4,994
		ispd18_test9	54,757,304	2,409,974	20	8.4	468	114	602	2,063
		ispd18_test10	68,123,435	2,602,267	576	232.5	596	639	1,811	9,965
		Geo. Mean Ratio	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	Dr. CU + FastPass	ispd18_test1	440,604	37,442	0	0.0	0	0	**0**	9
		ispd18_test2	7,929,759	376,531	0	0.0	0	1	**1**	68
		ispd18_test3	8,817,839	380,749	11	1.5	0	6	**17**	1,545
		ispd18_test4	26,394,567	760,959	13	10.9	112	60	185	2,254
		ispd18_test5	27,874,102	942,666	3	2.8	125	92	**220**	2,009
		ispd18_test6	35,821,552	1,446,618	0	0.0	200	99	299	1,239
		ispd18_test7	65,338,312	2,346,860	9	3.7	237	157	403	4,035
		ispd18_test8	65,622,748	2,357,369	4	1.7	265	152	421	3,936
		ispd18_test9	54,921,494	2,352,642	0	0.0	263	174	437	2,435
		ispd18_test10	68,461,622	2,560,944	2,626	1,494.0	400	1,173	4,199	20,618
		Geo. Mean Ratio	1.006	1.039	0.079	0.125	0.809	0.309	0.276	0.883
CUGR Guides	Dr. CU	ispd18_test1	428,861	35,213	1	0.1	0	2	3	13
		ispd18_test2	7,797,847	364,689	1	0.1	0	89	90	307
		ispd18_test3	8,763,379	361,199	144	41.3	0	125	269	731
		ispd18_test4	26,351,827	727,315	184	33.0	15	643	842	1,777
		ispd18_test5	27,519,172	927,075	345	63.2	77	443	865	6,482
		ispd18_test6	35,588,897	1,387,512	2	0.8	119	628	749	1,127
		ispd18_test7	64,897,135	2,289,543	672	129.6	161	92	925	6,817
		ispd18_test8	65,502,355	2,345,520	691	132.9	179	120	990	7,516
		ispd18_test9	54,361,528	2,341,329	3	1.1	177	105	**285**	2,958
		ispd18_test10	68,099,360	2,495,210	50	22.9	222	563	835	6,361
		Geo. Mean Ratio	0.996	1.007	0.587	0.468	0.498	0.996	0.863	1.094
	Dr. CU + FastPass	ispd18_test1	435,226	40,851	0	0.0	0	1	1	12
		ispd18_test2	7,894,401	419,344	0	0.0	0	49	49	312
		ispd18_test3	8,862,645	419,379	0	0.0	0	50	50	582
		ispd18_test4	26,391,398	765,580	3	3.3	12	33	**48**	1,511
		ispd18_test5	27,608,938	888,614	7	5.3	117	113	237	3,904
		ispd18_test6	35,704,772	1,320,632	2	0.5	190	84	**276**	2,148
		ispd18_test7	65,060,964	2,170,206	2	0.5	206	128	**336**	7,377
		ispd18_test8	65,645,086	2,223,406	3	0.8	165	156	**324**	7,783
		ispd18_test9	54,521,667	2,225,218	1	0.6	205	126	332	3,928
		ispd18_test10	68,297,974	2,369,604	39	18.7	226	508	**773**	6,101
		Geo. Mean Ratio	1.002	1.026	0.032	0.047	0.553	0.465	0.311	1.097

* Wire Length Unit: M2 pitch; Short Area Unit: Square of M2 pitch; # MAR: the number of min-area rule violations; Total DRC #: the summation of the number of short, the number of min-area rule violations, and the number of spacing rule violations. When calculating the geometric mean ratio, we first calculate the geometric mean of ten cases and do division. Since there exist zero values, we add one to each entry before calculating the geometric mean and then subtract one from the result.



(a) *Short area.*



(b) *Total DRC violation number.*

Figure 4.12: *Detailed routing result comparison with different maximum rip-up and reroute iteration number for ispd*18*_test*5*.*

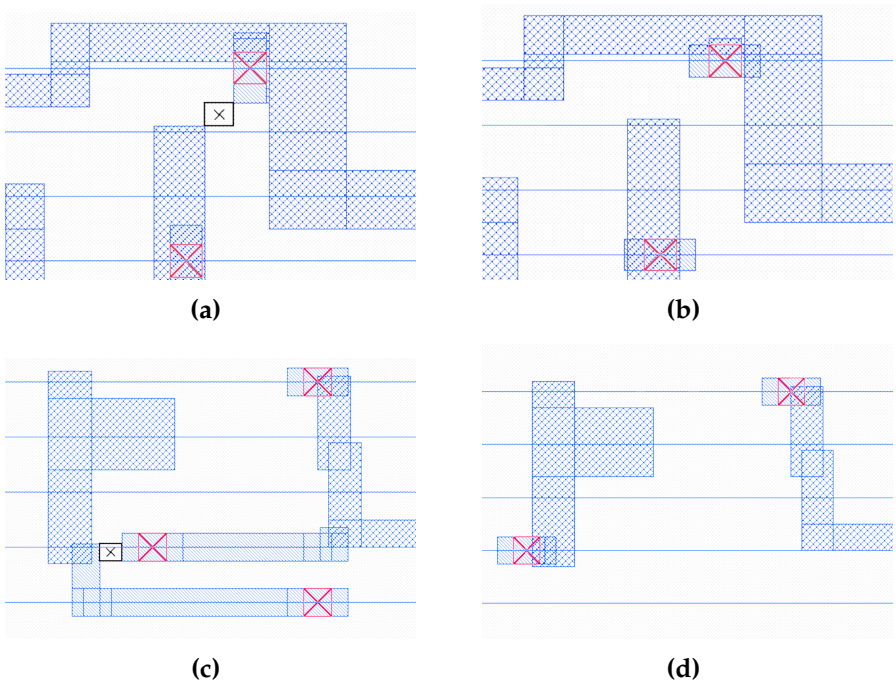


Figure 4.13: *Pin Access Results from ispd18_test4. (a) and (c) show the pin access-related DRC violations produced by Dr. CU. (b) and (d) show the DRC-clean pin access solution from FastPass.*

detailed router with a DRC-clean pin access solution with little runtime overhead.

We notice that the original Dr. CU outperforms FastPass in *ispd18_test10* with shorter runtime and fewer DRC violations when the ISPD 2018 guides are used. This can be caused by the congestion in the provided global routing result since the quality of route guides from the contest benchmark suite can have a large variance across different cases. Therefore, we generate another set of route guides with CUGR [79], one of the state-of-the-art LEF/DEF-based global routers, to verify the effectiveness of FastPass. In the lower part of Table 4.4, we show that with CUGR guides, FastPass consistently outperforms Dr. CU with 89.9% less short area and 63.9% fewer DRC violations. It is also able to produce the best result on *ispd18_test10* compared with all other three settings.

4.4 Summary

In this chapter, we present a fast and robust pin access analysis framework called FastPass for detailed routability enhancement. The framwork incorporates a sweep-line style algorithm for efficiently design rule checking and an incremental SAT-based algorithm for correct-by-construction route selection. Experimental results on the ISPD 2018 contest benchmark suite demonstrate that the proposed framework can consistantly achieve DRC-clean pin access schemes across all cases while being an order of magnitude faster than the state-of-the-art approach. We further integrate FastPass into the open-source detailed router Dr. CU. With the integration of FastPass, the detailed router is able to produce significantly better results with 87.5% less short area and 72.4% fewer DRC violations, which demonstrates the effectiveness of our proposed approach.

Chapter 5

Security-Aware Physical Design

In this chapter, we proactively and systematically harden the physical layouts of integrated circuits (ICs) against post-design insertion of Trojans. Our main contributions are summarized as follows.

- A layout-level multiplexer (MUX)-based logic locking scheme, called *TroMUX*, is proposed to hinder post-design Trojan insertion. TroMUX is devised to withstand state-of-the-art, machine learning (ML)-based attacks on locking, such as SCOPE [4] and MuxLink [6]. This is essential to hinder adversaries from circumventing the Trojan defense. Additionally, by the security promise of locking, attackers cannot easily insert targeted Trojans anymore, i.e., Trojans that require understanding of the original design. Furthermore, by densely filling up the layout with as many TroMUX instances as practically possible, attackers cannot easily insert additional Trojan logic in general anymore.

- An effective and efficient methodology for security closure against post-design Trojan insertion is proposed. The methodology is fully integrated into a commercial-grade physical-synthesis flow and provides a good control of security-versus-overheads trade-offs.

The rest of the chapter is organized as follows. In Section 5.1, we introduce threat models considered in this thesis. Our proposed algorithms will be discussed in Section 5.2 and the experimental results will be shown in Section 5.3.

5.1 Threat Models

5.1.1 Trojans

We follow a classical threat model as follows.

(1) Adversaries reside within manufacturing facilities, and we consider the design process to be trustworthy.

(2) As a consequence of the above assumption, adversaries only have knowledge of the protected physical layout, not the original unprotected design.

(3) Since we assume adversaries are within manufacturing facilities, we do not consider Trojans introduced by malicious designers or third-party IP modules.

(4) Trojans are implemented using regular standard cells, not by modifying interconnects or transistors.

(5) Targeted Trojans utilize triggers based on LCNs and payloads that target security-critical components such as key registers.

(6) The objective of the adversaries is to insert Trojan components, specifically trigger and payload components, into the layout.

5.1.2 Locking

First, recall that we employ locking not for IP protection, but to hinder Trojan insertion. Toward that end, we follow an oracle-less threat model that is consistent with both the literature and the above threat model on Trojans as follows.

(1) The adversaries' objective is to circumvent the locking scheme, to (i) understand the true functionality of the design, required for targeted Trojan insertion, and (ii) to regain layout resources, required for Trojan insertion in general.

(2) Given that adversaries are assumed to reside within manufacturing facilities, only oracle-less attacks are applicable. None of the well-known Boolean satisfiability-based attacks, including those tackling scan chains, are applicable.

(3) Following Kerckhoffs' principle, all implementation details for TroMUX locking are known to the adversaries; only the key-bits remain undisclosed.

5.2 Methodology

This work is motivated by the need for a proactive, pre-silicon Trojan-prevention scheme that is robust, effective, and efficient.

Our approach, which can be summarized as locking and layout filling applied in unison, aims to hinder first-order and second-order Trojan-insertion attacks. We integrate an ML-resilient, MUX-based locking scheme directly into the IC layouts. Unlike prior art, we neither employ trivial filler/spare cells, nor separate circuitry, nor vulnerable locking schemes. To protect IC layouts holistically, our methodology carefully embeds, in a security-aware manner, as many locking instances during physical instances as practically possible, i.e., keeping the design quality well under control. Our methodology is fully integrated into commercial-grade design tools.

Next, we describe the components of our methodology, i.e., a MUX-based locking scheme and a physical-synthesis flow for security- and design-aware locking and layout filling. Note that both are devised to be working in unison, but also require individual solutions toward that end.

5.2.1 TroMUX

Our MUX-based locking scheme, TroMUX, is specifically devised to hinder post-design Trojan insertion. Given the rise of powerful ML-based attacks on simple locking schemes, e.g., on X(N)OR key-gates [7], we opt for MUX-based locking which is considered more resilient [110]. Still, even MUX-based schemes have been attacked recently [6, 4]. To render TroMUX resilient against such advanced ML-based attacks, we devise the following implementation.

(a) Locking Approach: Unlike prior art, TroMUX is not based on obfuscating the netlist connectivity through MUX key-gates, but on using MUX key-gates for regular locking. Thus, for TroMUX, there is no information leakage arising from MUX key-gates for the connectivity and structure of the design, which is the key vulnerability of prior art [6]. Instead, TroMUX employs simple but resilient key-gate structures with localized connectivity, described next.

(b) Learning-Resilient Key-Gate Structures: The design of TroMUX ensures that key-bits are fully randomized, without any correlation to each of the locked gate's true functionality. To do so, TroMUX instances render the locked gates interchangeable with respect to their complementary counterparts, e.g., a NAND gate can act as AND or NAND, only depending on the key-bit.

When locking gates using TroMUX instances, one picks randomly from the

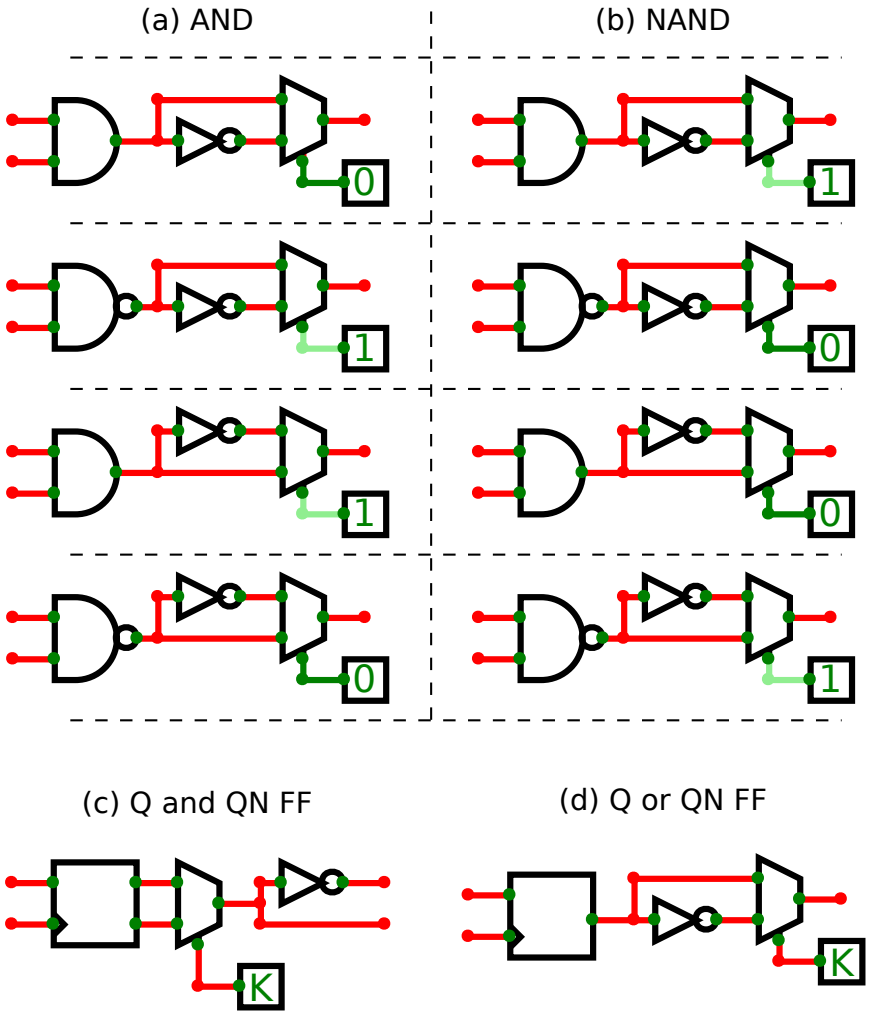


Figure 5.1: *Design of different TroMUX instances. The key-bit is connected to the MUX select line. (a, b) Locking of simple AND, NAND gates. The four rows show TroMUX instances which are pairwise structurally equivalent and thus functionally indistinguishable without knowing the key-bit. Note that, for 2/4 or half the respective TroMUX instances, the original AND, NAND gate is first transformed into its counterpart. This serves for further design obfuscation regarding the distribution of gate types. (Other simple gates are locked similarly; not illustrated here.) (c, d) Locking of flip-flops (FFs) with different output configurations. For (c), the key-bit dictates which output signal holds Q and which QN, whereas for (d), the key-bit dictates whether Q or QN is put out. For (d), the original Q/QN FF can also be transformed to its QN/Q counterpart (not illustrated here).*

different possible configurations (Figure 5.1). It is essential to note how the different configurations are structurally indistinguishable; only the key-bits determine the true functionality. Also, key-bits are easy to randomize for any particular true functionality, simply by randomly picking one of the possible configurations. For simple complementary gates, like AND (Figure 5.1[a]) and NAND (Figure 5.1[b]), there are four different pairwise configurations which are indistinguishable on their own. Note how half the configurations are based on first transforming the original gates to their counterparts; this serves for further obfuscation of the overall layout regarding the distribution of gate types.

(c) Locking of Complex Gates: Concerning flip-flops (FFs), these can be locked as is, by connecting the Q and QN ports to a dedicated key-gate structure (Figure 5.1[c])[1]. For commercial libraries, where FF outputs are typically optimized and trimmed, similar key-gate structures as for simple gates can be initiated (Figure 5.1[d]).

For other complex gates like AOI, we observe that complementary counterparts appear rarely in an optimized layout, if they are available at all in the library. Thus, we defer locking of such complex gates as follows. When such complex gate is selected for locking, we search its fan-out in depth-first manner for simple gates and FFs, covering all the fan-out paths. Then, all the found simple gates and FFs are locked instead, which essentially translates to locking the downstream structure of that complex gate.

(d) No Information Leakage from Physical Layout: As indicated, there are no direct correlations between the types of original versus the locked gate, the connectivity within and across TroMUX instances, and the correct key-bit; all are interchangeable,

[1]We note that, independently of our work, this key-gate structure was proposed by Karmakar et al. [65], though in the context of locking scan chains.

randomly chosen, and thus indistinguishable for an attacker.

It is also important to note that the two internal TroMUX nets connecting to the MUX inputs (or driving the MUX fan-out for the case of Q and QN FFs, Figure 5.1[c]) share a path and are, thus, both optimized/impacted at once by EDA tools. Accordingly, an attacker seeking to study the underlying timing paths, driver strengths, etc., would not gain any additional information.

5.2.2 Physical Synthesis

With the proposed locking scheme, we introduce a physical-synthesis flow that is capable of hardening any post-route layout with negligible timing and area overheads. As outlined in Figure 5.2, we start with the original netlist and other necessary data to generate a baseline layout and then carefully interleave logic locking and physical synthesis to produce a final protected layout.

(a) Two-Stage Locking Scheme: In the first stage, we lock all security-critical FFs, as defined by assets. After locking, we conduct placement and routing (P&R), and obtain a partially protected layout (PPL). A PPL has only FF assets protected; no other parts like LCNs and LCCs (i.e., the cells driving LCNs; both are targets for Trojan triggers) are protected yet. Further, a PPL has relatively low utilization, leaving placement sites exposed to Trojan insertion.

In the second stage, we thus aim at locking as many cells as possible to fill the layout, also prioritizing the locking of LCCs, all without degrading the design quality. With the physical layout-related information obtained from the PPL, we first derive the number of cells to be locked, then perform cell selection considering timing and controllability (detailed in Section 5.2.2). After locking all selected cells, P&R is re-run, and we finally retrieve a highly-utilized layout with all security-

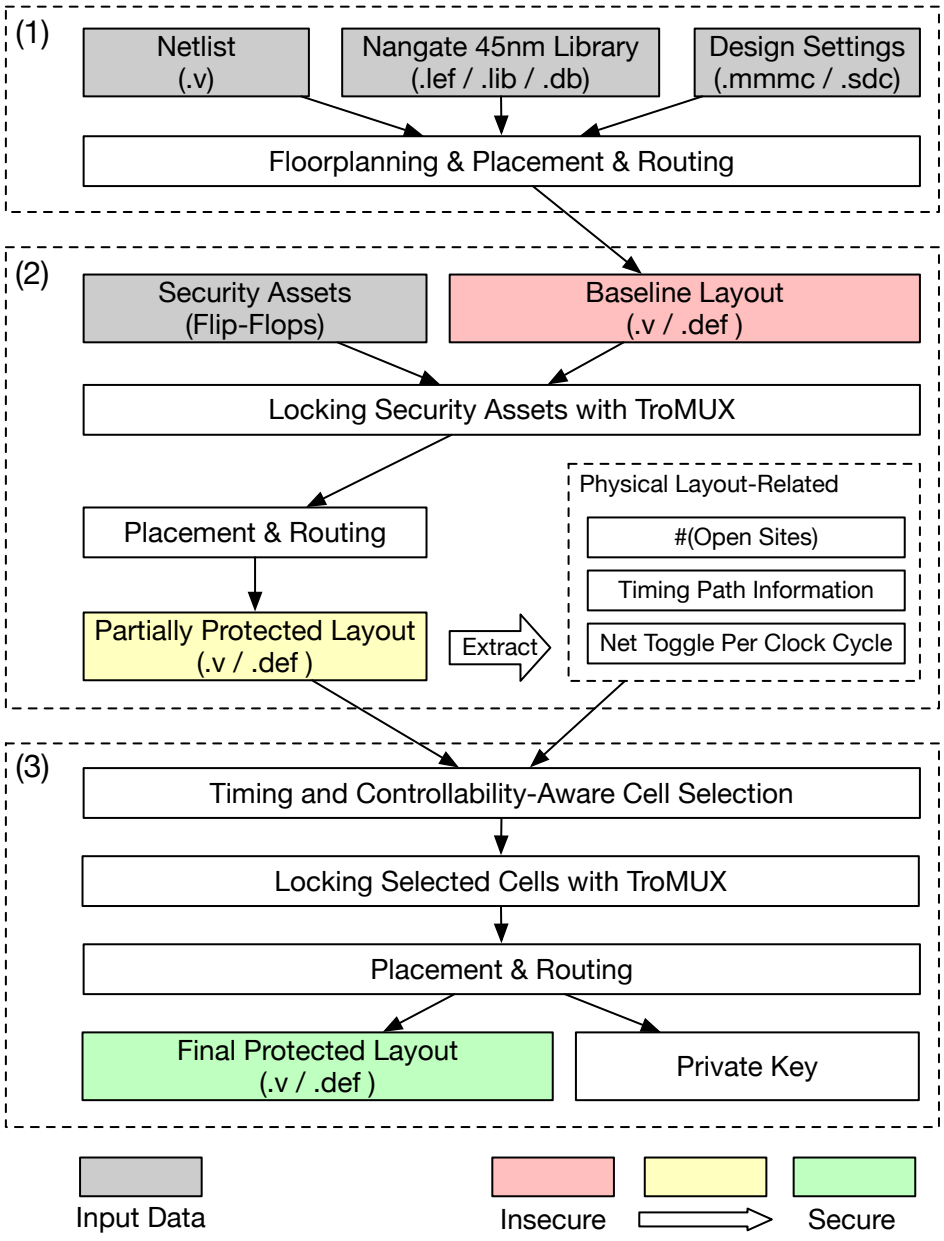


Figure 5.2: *Our physical-synthesis flow, which consists of three parts: (i) initial synthesis, (ii) locking of security assets, and (iii) locking considering timing and controllability.*

critical components well protected through both TroMUX locking and much fewer placement sites left open.

(b) Storage of Key-Bits: For each TroMUX instance, there will be one corresponding key-bit, requiring some facility to store all key-bits. We employ a large shift register for the following reasons:

- Prior art often considers adding a dedicated PI for each key-bit, which is not scalable. For us, we only require two additional PIs (data in, load) for any number of key-bits.
- Ours incurs negligible impact on performance and power,[2] while the impact on area is even desired (see below).
- The FFs used to build up the shift register are also helpful for locally filling open placement sites as needed, whereas bulky memory blocks would rather complicate this task.
- Finally, memory blocks are unavailable for the library used in this work, i.e., Nangate 45nm Open Cell Library [94]. We require this library for the recently introduced ISPD 2022 benchmarks for the contest on security closure of physical layouts [68].

(c) On-Demand Key Length: TroMUX instances occupy open placement sites with their INVs, MUXes, and FFs. After locking security assets, we determine the

[2]For a design locked with k key-bits, loading will take k clock cycles. This is done during initial boot-up, when the main circuitry is still hold in reset; runtime cost for such one-time initialization are considered negligible. Once the load signal is set low, the key-bits will remain stable, consuming only some static FF power.

additional key length required to fill the physical layout at hand as follows:

$$k = floor\left(\frac{num\_open\_sites}{size(INV) + size(MUX) + size(FF) + \alpha}\right) \tag{5.1}$$

where $size(INV)$ represents the size of the smallest INV cell in the library, etc., and α is a parameter for timing budget. As commercial tools will usually conduct timing optimization by gate sizing and buffer/inverter insertion, we need to reserve some additional space during locking. Accordingly, for designs where timing closure is more challenging, we will need a larger α, and vice versa.

(d) Cell Selection Considering Timing and Controllability: Recall that, in the second locking stage, we lock selected cells to reduce the number of open sites and protect more LCNs/LCCs. As key-gate structures can introduce further cell delays to related timing paths, the selection of cells to lock becomes critical for timing closure. Thus, we propose a scoring function $cellScore(c, N, P)$ to comprehensively describe the priority of a cell c as follows.

$$cellScore(c, N, P) = \sum_{n \in N(c)} netScore(n, P), \tag{5.2}$$

$$netScore(n, P) = \frac{1}{1 + exp(-2 \cdot MS(n, P))} \cdot \frac{1}{TPC(n) + 10^{-3}}, \tag{5.3}$$

$$MS(n, P) = \begin{cases} \min\limits_{p \in P(n)} p.slack & \text{, if } |P(n)| > 0, \\ -0.5 & \text{, otherwise,} \end{cases} \tag{5.4}$$

with terms described in Table 5.1. The cell score is represented as the sum of net scores to generalize to multi-output cases, e.g., both Q and QN of a FF are used. Further, we devise $netScore(n, P)$ to jointly consider timing and controllability. Using sigmoid, the score value remains positive even for negative but small slacks. Besides, since $TPC(n) \in [0, 2]$ and nets with $TPC \leq 0.1$ are considered as LCNs in

Table 5.1: *Notations for Cell Selection*

Term	Description
C	The set of standard cell instances
c	A cell instance from C
N	The set of nets
n	A net from N
$N(c)$	The set of nets driven by c
P	The set of timing paths
$P(n)$	The set of timing paths covering n
$MS(n, P)$	The minimum slack of paths covering n
$TPC(n)$	The number of toggles per clock cycle for n

this work, we add a small margin (10^{-3}) to avoid both div-by-0 and $TPC(n)$ from dominating the score. When calculating MS, some nets may not be covered by any reported timing paths (due to limitations of the commercial tool). For those nets, $MS(n, P)$ returns -0.5, a fall-back value close to average negative slacks observed; this value serves to conservatively penalize cells with unknown timing.

Using the above scoring, we propose an iterative cell-selection algorithm in Algorithm 8. In each iteration, we calculate cell scores (line 3–4) and pick those cells with the highest score (line 5–7). Next, we update the slacks of affected timing paths based on σ, a pessimistic estimate of delay introduced by locking, defined as the sum of worst-case delays for INV_X1 and MUX2_X1 (i.e., the default TroMUX cells for our experiments), as derived from the libraries for the matching corner cases (line 8–10).

Our initial experiments show that the proposed two-stage locking scheme is

Algorithm 8 Cell Selection Considering Timing and Controllability

Input: Standard Cells C, Nets N, Timing Paths P, Key Length K, Locking Delay σ.
Output: The Set of Cells to Lock C'.

$C' \leftarrow \{\}$;
while $|C'| < K$ **do**
  foreach cell $c \in C$ **do**
    $c.score \leftarrow cellScore(c, N, P)$; ▷ Score calculation for each cell
  Let c_h be the cell with the highest score in C;
  $C'.append(c_h)$;
  $C.remove(c_h)$;
  foreach net $n \in N(c_h)$ **do**
    foreach timing path $p \in P(n)$ **do**
      $p.slack \leftarrow p.slack - \sigma$; ▷ Pessimistic slack estimation
return C';

more timing-friendly over a single-stage approach. That is, when we did directly use the physical information extracted from the baseline layouts (marked in red in Figure 5.2) for cell selection and related locking, it was much harder to achieve timing closure, due to the accumulation of slack estimation errors.

5.3 Experimental Results

5.3.1 Setup

(a) Tools: As indicated, we base our work on a commercial-grade design flow. Without loss of generality, we use Cadence Innovus 20.14 for physical synthesis. The methodology is implemented in custom TCL scripts and Python code. For security analysis using SCOPE and MuxLink, setup details are provided in Section 5.3.3.

(b) Benchmarks: We employ the benchmark suite from the ISPD 2022 Contest on Security Closure [68].

The suite comprises a range of crypto cores as well as the *openMSP430* microcon-

Table 5.2: *Benchmark Statistics*

Design	F. Utils	#(Cells)	#(Nets)	#(ML)	CP	Corner
AES_1	75.0%	16509	19694	10	1	typical
AES_2	75.0%	16509	19694	10	1	typical
AES_3	85.0%	15836	19020	10	1	typical
Camellia	50.0%	6710	7160	6	10	slow
CAST	50.0%	12682	13057	6	10	slow
MISTY	50.0%	9517	9904	6	10	slow
openMSP430_1	50.0%	4690	5312	6	30	slow
openMSP430_2	70.0%	5921	6550	6	8	slow
PRESENT	50.0%	868	1046	6	10	slow
SEED	50.0%	12682	13057	6	10	slow
SPARX	50.0%	8146	10884	6	10	slow
TDEA	70.0%	2269	2594	6	4	slow

* F. Utils: floorplanning utilization for resynthesis; #(Cells/Nets): nr. of cells/nets in original netlists; #(ML): nr. of metal layers; CP (ns): clock period; Corner: typical is characterized for 1.1V and 25C, slow for 0.95V and 125C.

troller. As Table 5.2 shows, the designs vary in terms of complexity, utilization, size (cells and nets), available metal layers, timing constraints, and corners. Since the suite was synthesized by legacy versions of Cadence Innovus, we resynthesize all designs at our end with floorplan utilization rates similar to the original post-route benchmark layouts; only the rates for *AES_3*, *openMSP430_2*, and *TDEA* are set 10% lower, as needed to lock all security assets.

5.3.2 Results on the ISPD 2022 Contest Benchmarks

We quantify security and layout results for the resynthesized baseline layouts versus the final, protected layouts in Table 5.3.

First, all the protected layouts show much higher utilization rates, thereby reduc-

Table 5.3: *Layout and Security Results for Ours on the ISPD 2022 Contest Benchmark Suite*

Design	Baseline Layout (Resynthesized)						Protected Layout (Final)									
	Utils	#(Open)	TU	WNS	TNS	Power	Utils	#(Open)	Δ(Open)	TU	WNS	TNS	Power	#(LSA)/#(SA)	#(LLCC)/#(LCC)	KL
AES_1	75.4%	43,980	8.7%	0.000	0.000	59.957	96.2%	6,838	-84.5%	11.1%	-0.013	-0.043	60.552	291/291	884/1,389	1,199
AES_2	75.1%	44,420	8.7%	-0.001	-0.003	59.441	96.3%	6,649	-85.0%	10.9%	-0.008	-0.017	61.040	291/291	908/1,431	1,202
AES_3	85.7%	22,129	9.7%	-0.001	-0.002	59.869	96.6%	5,225	-76.4%	11.2%	-0.031	-4.657	62.787	291/291	150/1,310	441
Camellia	49.5%	33,919	9.5%	1.194	0.000	1.233	98.9%	753	-97.8%	13.3%	0.015	0.000	1.488	256/256	724/724	1,271
CAST	49.1%	54,444	9.1%	0.047	0.000	3.136	93.6%	6,879	-87.4%	12.7%	-0.134	-1.455	3.912	192/192	983/994	1,572
MISTY	48.5%	43,359	8.6%	-0.021	-0.037	2.238	94.4%	4,686	-89.2%	12.4%	-0.140	-1.515	2.966	204/204	307/312	1,215
openMSP430_1	49.7%	32,799	6.2%	0.000	0.000	0.375	97.9%	1,389	-95.8%	12.6%	0.000	0.000	0.544	340/340	441/456	1,218
openMSP430_2	70.5%	15,125	7.7%	0.000	0.000	1.239	97.1%	1,497	-90.1%	10.6%	-0.006	-0.015	1.369	334/334	209/696	543
PRESENT	49.8%	6,284	4.4%	6.694	0.000	0.198	98.0%	245	-96.1%	6.8%	4.935	0.000	0.235	80/80	3/3	241
SEED	49.1%	54,444	9.1%	0.047	0.000	3.136	94.3%	6,056	-88.9%	12.7%	-0.182	-1.072	3.908	195/195	979/989	1,612
SPARX	49.9%	69,979	6.3%	2.452	0.000	2.164	98.8%	1,658	-97.6%	14.0%	0.027	0.000	2.822	2,176/2,176	361/375	2,582
TDEA	70.4%	5,559	6.8%	0.049	0.000	1.084	98.4%	304	-94.5%	8.0%	0.027	0.000	1.153	168/168	6/47	214

* Utils: util. after phys. synthesis; #(Open): nr. of open sites; TU: track utilization; WNS (ns): worst negative slack; TNS (ns): total negative slack; Power (mW): total power; Δ(Open): reduction of nr. of open sites; SA: security assets; LSA: locked security assets; LCC: low-controllable cells; LLCC: locked low-controllable cells; KL: key length (bits).

ing open placement sites by 90.3% on average and rendering designs much more resilient against Trojan insertion in general. Meanwhile, we achieve higher track utilization (defined as *total routed wire length / total track length*) compared with the baseline layout such that routing from triggers to payloads will be more challenging for the attackers. Second, as another layer of protection, recall that all security assets are locked and that LCNs/LCCs are well locked in most layouts, except for those with relatively high initial utilizations. Third, all layouts are without any DRC violations, despite the ultra-high utilization of 93.6–98.9%[3]. This demonstrates the effectiveness of our proposed flow. Fourth, total power is increased by 18.5% on average, which seems reasonable given all the additional cells introduced with TroMUX instances.

In Figure 5.3, we show the layouts of the exemplary *Camellia* design in three

[3] In general, achievable utilization is subject to the design and even more to the technology library. For the latter, we note that the Nangate 45nm Open Cell Library [94], used in this work for the ISPD 2022 contest benchmarks [68], has rather loose constraints and design rules, enabling high utilization. For our work, showcasing to reach such ultra-high utilization is important—it supports our claim to truly protect layouts against Trojan insertion. As indicted, prior art like [122] is challenged by high utilization.

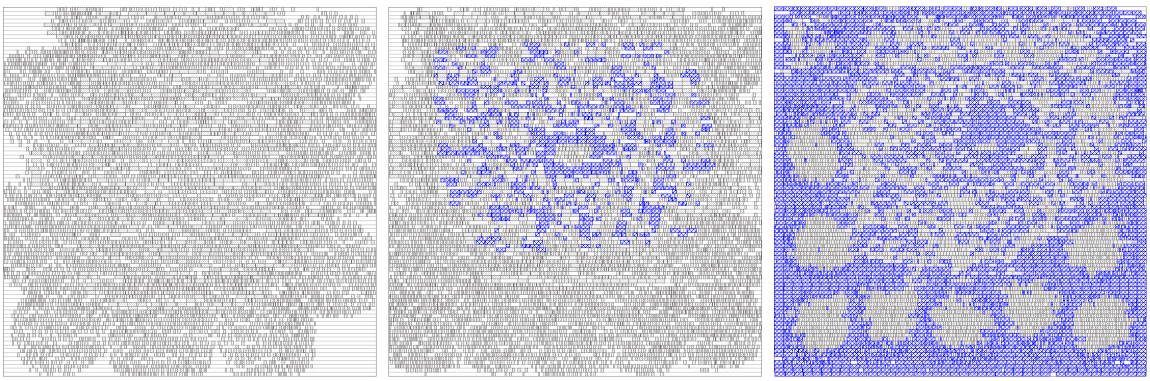


Figure 5.3: *Camellia layout, after initial synthesis (left), locking of security assets (middle), and locking considering timing and controllability (right). The utilization increases from 49.5% to 59.2%, and eventually to 98.9% (see also Footnote 3). Gates introduced by TroMUX instances are marked in blue while the other standard cells are filled with grey dots.*

stages as described in Figure 5.2.

5.3.3 ML-Based Attack Analysis

Recall that we use logic locking to both protect security-critical components in particular and the layout in general. Attackers would want to undermine our locking and remove TroMUX instances, to be able to insert Trojans targeted on assets and LCNs/LCCs in particular (Section 5.1). Thus, we evaluate TroMUX against state-of-the-art ML-based attacks SCOPE [4] and MuxLink [6].

Setup for SCOPE: The attack expects designs in *bench* format. Thus, we resynthesize our locked benchmarks using the *bench*-specific, limited set of cells, (i.e., AND, OR, NAND, NOR, INV, BUF, DFF, XOR, and XNOR) and convert the netlists into *bench* format using an in-house *Python* script. Since the attack cannot handle loops in the locked designs, we represent FFs as pseudo PIs/POs. Further, we use the default margin value of 0 [4].

Table 5.4: *ML-Based Attack Results on Different Locking Schemes*

Design	KL	SCOPE [4]						MuxLink [6]							
		COPE (%)	TroMUX Key 1		TroMUX Key 2		#(X)	TroMUX				D-MUX [6, 110]			
			AC (%)	KPA (%)	AC (%)	KPA (%)		AC (%)	PC (%)	KPA (%)	#(X)	AC (%)	PC (%)	KPA (%)	#(X)
AES_1	1,199	0.1919	28.86	48.39	30.78	51.61	484	28.77	71.48	50.22	512	78.15	79.90	79.54	21
AES_2	1,202	0.1864	31.53	52.71	28.29	47.29	483	28.12	75.04	52.98	564	80.45	81.70	81.47	15
AES_3	441	0.1075	23.36	48.13	25.17	51.87	227	8.16	90.02	45.00	361	88.44	89.12	89.04	3
Camellia	1,271	0.0240	14.16	51.72	13.22	48.28	923	26.12	73.80	49.92	606	90.64	92.60	92.46	25
CAST	1,572	0.0359	18.58	50.52	18.19	49.48	994	37.47	66.03	52.45	449	93.74	94.32	94.29	7
MISTY	1,215	0.1036	23.70	48.90	24.77	51.10	626	33.33	65.84	49.39	395	97.84	98.03	98.02	3
openMSP430_1	1,218	0.0432	19.87	49.90	19.95	50.10	733	23.56	76.60	50.17	646	NA	NA	NA	NA
openMSP430_2	543	0.0306	29.28	52.13	26.89	47.87	238	9.39	90.42	49.51	440	66.85	69.98	69.01	17
PRESENT	241	0.0434	2.49	42.86	3.32	57.14	227	11.62	95.02	70.00	201	NA	NA	NA	NA
SEED	1,612	0.0343	18.55	49.10	19.23	50.90	1,003	37.03	64.02	50.72	435	97.33	97.70	97.70	6
SPARX	2,582	0.0176	21.42	50.23	21.22	49.77	1,481	5.38	94.93	51.48	2,312	NA	NA	NA	NA
TDEA	214	0.0001	0.47	100.00	0.00	0.00	213	1.87	99.07	66.67	208	NA	NA	NA	NA
Avg.	-	0.0682	19.35	53.72	19.25	46.28	-	20.90	80.19	53.21	-	86.68	87.92	87.69	-

* AC: accuracy; PC: precision; KPA: key prediction accuracy; COPE: COnstant Prop. Effect; #(X): nr. of undeciphered key-bits; NA: Locking using the script in [6] fails.

Setup for MuxLink: The original implementation supports only selected gates (i.e., AND, NAND, OR, NOR, INV, XOR, XNOR, BUF). Since we use the ISPD 2022 contest benchmarks where all gates from the Nangate library are used, we extend the one-hot feature vectors of MuxLink accordingly. Consistent with the original operation of MuxLink, we treat each pin of each gate as a node and the connections between pins, including those going through gates, as edges. The connections between the input/output pins of TroMUX key-gates are kept as test set for prediction, while all others (except PI or PO connections) are used as training set. We adopt the same graph neural network configuration and training hyperparameters as in [6]; in particular, number of hops for extracting subgraphs is set to 3 and the threshold in post-processing is set to 0.01.

Evaluation Metrics: We report the performance of the SCOPE and MuxLink attacks using the following established metrics: *accuracy* (AC), *precision* (PC), *key prediction*

accuracy (KPA), and *COnstant Propagation Effect* (COPE, [4]);[4] all metrics are in percentage.

Results for SCOPE: The results in Table 5.4 show that 19.35%/19.25% of the key bits are correctly recovered on average. The average value of 0.065% for COPE means that the attack fails almost entirely. Further, the average KPA of 53.72% versus 46.28% for the two keys[5] indicates that SCOPE is forced to random guessing. Note that the attack can only decipher a single bit for *TDEA*, thus resulting in KPA of 100% and 0% for that design.

Results for MuxLink: The results in Table 5.4 show that MuxLink can only predict, on average, 20.90% of the key-bits. Moreover, the average KPA is 53.21%, clearly indicating that MuxLink is forced to random guessing. In contrast to prior art like D-MUX [110, 6], also showcased in Table 5.4, TroMUX is superior in thwarting the attack.

5.3.4 Discussion of Prior Art

Recall the review of related prior art in Section 2.3. Note that none of the prior art released their results artifacts publicly. Further, most of the prior art uses different technology libraries and implementation schemes, as well as different benchmarks. Thus, a direct comparison is impractical.

[4]AC measures the percentage of correctly deciphered key-bits, i.e., $(k_{correct}/k_{total})$. PC measures the correctly deciphered key-bits, optimistically considering every X/undeciphered value as a correct guess, i.e., $((k_{correct}+k_X)/k_{total})$. KPA measures the correctly deciphered key-bits over the entire prediction set, i.e., $(k_{correct}/(k_{total}-k_X))$. COPE measures the vulnerability against the SCOPE attack; COPE = 0% means the attack fails entirely.

[5]SCOPE predicts the two mutually complementary keys, representing the possible configurations of '0' and '1' versus '1' and '0' for key-bits.

Still, it is essential to recall the considerable limitations of those studies and to forecast related implications. For example, for X(N)OR key-gates utilized by Samimi et al. [106] and Marcelli et al. [88], we note that Alrahis et al. [7] have independently shown that such locking can be circumvented with up to 97.22% accuracy. Thus, we argue that locking as applied by Samimi et al. [106] and Marcelli et al. [88] can be easily circumvented, unlike TroMUX (Section 5.3.3).

5.4 Summary

In this chapter, we propose a MUX-based locking scheme, called TroMUX, along with a carefully tuned physical-synthesis flow, to prevent targeted Trojan insertion. To the best of our knowledge, our approach is the first to systematically combine locking and layout-level techniques for effective Trojan prevention. Experimental results on the ISPD 2022 contest benchmarks show that our approach can reduce the number of open placement sites by 90.3% on average, with security-critical components secured by logic locking. In addition, we demonstrate the superior resilience of our approach against ML-based attacks. Experimental results show that both SCOPE and MuxLink achieve an average key-prediction accuracy of approximately 50%, which indicates that our scheme effectively prevents key prediction, leading to random guessing as the primary outcome.

Conclusion

In this thesis, we propose a set of algorithms for improved routability and security in physical layout. Regarding routability, we study two critical problems in VLSI routing. First, we address the misalignment between the placement and routing (P&R) objectives with a P&R co-optimization engine called Starfish, which conducts cell movement and net rerouting iteratively to minimize the routed wire length without causing any overflow. Starfish adopts a lookup table (LUT)-based approach for efficient and accurate wire length estimation under various routing constraints. Based on the LUT, we propose a cell movement scheme with A* search-based partial rerouting, which is further accelerated with a median box-based pruning technique. As a result, Starfish is able to significantly reduce the total wire length of an initial P&R solution without causing any overflow. To further enhance solution quality, future work can explore the simultaneous movement of multiple cells.

We then focus on detailed routing and propose a fast pin access analysis framework (PAAF) called FastPass for enhancing detailed routability. FastPass incorporates a sweep-line style algorithm for efficiently design rule checking and an incremental SAT-based algorithm for correct-by-construction route selection. Experimental results show that the proposed framework can consistently achieve DRC-clean pin access schemes while being an order of magnitude faster than the state-of-the-art approach. With FastPass, Dr. CU can produce detailed routing re-

sults with much less short area and fewer DRC violations. Future work can explore dynamic pin access, where pin access analysis is conducted iteratively within the routing flow, adjusting the produced DRC-clean pin access scheme based on the actual routing demand or historical violations near the pins.

In addition to performance metrics, we also consider security as an emerging objective in physical design. To harden physical layouts against post-design insertion of hardware Trojans, we propose a multiplexer (MUX)-based locking scheme designed to withstand machine learning (ML)-based attacks on locking. Based on that, we propose a security-aware physical design flow capable of hardening any post-route layout while maintaining controllable timing and area overheads. Experimental results show that our approach significantly reduces the number of open placement sites, making Trojan insertion more challenging. Moreover, security-critical components are secured by the proposed logic locking scheme, which demonstrates superior resilience against the state-of-the-art ML-based attacks. Nevertheless, higher layout utilization and additional logics can increase parasitic capacitance and data path delay respectively, thus bringing more difficulties to timing and power optimization. Therefore, one possible future direction could aim at reducing the performance overhead of the defense mechanism.

By addressing the challenges in routability and security, this thesis contributes to the advancement of physical design methodologies. Specifically, we propose a P&R co-optimization framwork that effectively reduces the routed wire length in a given layout. Additionally, we introduce a fast pin access analysis framework that enhances the detailed-routability of a detailed router. Furthermore, we present various techniques, including logic locking and layout filling, to improve the security of physical layouts. These contributions pave the way for better design quality and security in VLSI physical design.

References

[1] Nobel Media AB 2014. *The History of the Integrated Circuit*. https://web.archive.org/web/20180629102838/https://www.nobelprize.org/educational/physics/integrated_circuit/history/.

[2] Erfan Aghaeekiasaraee, Aysa Fakheri Tabrizi, Tiago Augusto Fontana, Renan Netto, Sheiny Fabre Almeida, Upma Gandhi, José Luís Güntzel, David Westwick, and Laleh Behjat. "Cr&p: An efficient co-operation between routing and placement". In: *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2022, pp. 772–777.

[3] Markus Ahrens, Michael Gester, Niko Klewinghaus, Dirk Müller, Sven Peyer, Christian Schulte, and Gustavo Tellez. "Detailed routing algorithms for advanced technology nodes". In: *IEEE Transactions on computer-aided design of integrated circuits and systems* 34.4 (2014), pp. 563–576.

[4] Abdulrahman Alaql, Md Moshiur Rahman, and Swarup Bhunia. "SCOPE: Synthesis-based constant propagation attack on logic locking". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29.8 (2021), pp. 1529–1542.

[5] Charles J Alpert, Zhuo Li, Michael D Moffitt, Gi-Joon Nam, Jarrod A Roy, and Gustavo Tellez. "What makes a design difficult to route". In: *Proceedings of the 19th international symposium on Physical design*. 2010, pp. 7–12.

[6] Lilas Alrahis, Satwik Patnaik, Muhammad Shafique, and Ozgur Sinanoglu. "MuxLink: Circumventing learning-resilient MUX-locking using graph neural network-based link prediction". In: *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2022, pp. 694–699.

[7] Lilas Alrahis, Satwik Patnaik, Muhammad Shafique, and Ozgur Sinanoglu. "OMLA: An oracle-less machine learning-based attack on logic locking". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 69.3 (2021), pp. 1602–1606.

[8] Apple. *Apple introduces M2 Ultra*. https://nr.apple.com/DH9c4J9oT2.

[9] Papa-Sidy Ba, Sophie Dupuis, Manikandan Palanichamy, Marie-Lise Flottes, Giorgio Di Natale, and Bruno Rouzeyre. "Hardware trust through layout filling: A hardware Trojan prevention technique". In: *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE. 2016, pp. 254–259.

[10] Papa-Sidy Ba, Manikandan Palanichamy, Sophie Dupuis, Marie-Lise Flottes, Giorgio Di Natale, and Bruno Rouzeyre. "Hardware trojan prevention using layout-level design approach". In: *2015 European Conference on Circuit Theory and Design (ECCTD)*. IEEE. 2015, pp. 1–4.

[11] Shabbir Batterywala, Narendra Shenoy, William Nicholls, and Hai Zhou. "Track assignment: A desirable intermediate step between global routing and detailed routing". In: *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*. 2002, pp. 59–66.

[12] Bentley and Wood. "An optimal worst case algorithm for reporting intersections of rectangles". In: *IEEE Transactions on Computers* 100.7 (1980), pp. 571–577.

[13] Charles R Bonapace and C-Y Lo. "An O (nlogm) algorithm for VLSI design rule checking". In: *Proceedings of the 26th ACM/IEEE Design Automation Conference*. 1989, pp. 503–507.

[14] *Cadence Innovus Implementation System*. https://www.cadence.com/. Accessed: 2023-05-15.

[15] Prabuddha Chakraborty, Jonathan Cruz, and Swarup Bhunia. "SAIL: Machine learning guided structural analysis attack on hardware obfuscation". In: *2018 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE. 2018, pp. 56–61.

[16] Tony Chan, Jason Cong, and Kenton Sze. "Multilevel generalized force-directed method for circuit placement". In: *Proceedings of the 2005 international symposium on physical design*. 2005, pp. 185–192.

[17] Tony F Chan, Jason Cong, Tim Kong, Joseph R Shinnerl, and Kenton Sze. "An enhanced multilevel algorithm for circuit placement". In: *ICCAD-2003. International Conference on Computer Aided Design (IEEE Cat. No. 03CH37486)*. IEEE. 2003, pp. 299–306.

[18] Chin-Chih Chang and Jason Cong. "An efficient approach to multi-layer layer assignment with application to via minimization". In: *Proceedings of the 34th annual Design Automation Conference*. 1997, pp. 600–603.

[19] Fong-Yuan Chang, Ren-Song Tsay, Wai-Kei Mak, and Sheng-Hsiung Chen. "MANA: A shortest path maze algorithm under separation and minimum

length nanometer rules". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32.10 (2013), pp. 1557–1568.

[20] Yen-Jung Chang, Yu-Ting Lee, and Ting-Chi Wang. "NTHU-Route 2.0: A fast and stable global router". In: *2008 IEEE/ACM International Conference on Computer-Aided Design*. IEEE. 2008, pp. 338–343.

[21] Gengjie Chen, Chak-Wa Pui, Haocheng Li, and Evangeline FY Young. "Dr. cu: Detailed routing by sparse grid graph and minimum-area-captured path search". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39.9 (2019), pp. 1902–1915.

[22] Huang-Yu Chen, Chin-Hsiung Hsu, and Yao-Wen Chang. "High-performance global routing with fast overflow reduction". In: *2009 Asia and South Pacific Design Automation Conference*. IEEE. 2009, pp. 582–587.

[23] Tung-Chieh Chen, Zhe-Wei Jiang, Tien-Chang Hsu, Hsin-Chen Chen, and Yao-Wen Chang. "NTUplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27.7 (2008), pp. 1228–1240.

[24] Chung-Kuan Cheng, Andrew B Kahng, Ilgweon Kang, and Lutong Wang. "Replace: Advancing solution quality and routability validation in global placement". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38.9 (2018), pp. 1717–1730.

[25] Chris Chu and Yiu-Chung Wong. "FLUTE: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27.1 (2007), pp. 70–83.

[26] Lawrence T Clark, Vinay Vashishtha, Lucian Shifren, Aditya Gujja, Saurabh Sinha, Brian Cline, Chandarasekaran Ramamurthy, and Greg Yeric. "ASAP7: A 7-nm finFET predictive process design kit". In: *Microelectronics Journal* 53 (2016), pp. 105–115.

[27] Ke-Ren Dai, Wen-Hao Liu, and Yih-Lang Li. "NCTU-GR: Efficient simulated evolution-based rerouting and congestion-relaxed layer assignment on 3-D global routing". In: *IEEE Transactions on very large scale integration (VLSI) systems* 20.3 (2011), pp. 459–472.

[28] Ke-Ren Dai, Chien-Hung Lu, and Yih-Lang Li. "GRPlacer: Improving routability and wire-length of global routing with circuit replacement". In: *Proceedings of the 2009 International Conference on Computer-Aided Design*. 2009, pp. 351–356.

[29] Nima Karimpour Darav, Andrew Kennings, Aysa Fakheri Tabrizi, David Westwick, and Laleh Behjat. "Eh? Placer: A high-performance modern technology-driven placer". In: *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 21.3 (2016), pp. 1–27.

[30] Edsger W Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1 (1959), pp. 269–271. URL: http://eudml.org/doc/131436.

[31] Yixiao Ding, Chris Chu, and Wai-Kei Mak. "Pin accessibility-driven detailed placement refinement". In: *Proceedings of the 2017 ACM on International Symposium on Physical Design*. 2017, pp. 133–140.

[32] Chen Dong, Yi Xu, Ximeng Liu, Fan Zhang, Guorong He, and Yuzhong Chen. "Hardware trojans in chips: A survey for detection and prevention". In: *Sensors* 20.18 (2020), p. 5165.

[33] Sophie Dupuis, Papa-Sidi Ba, Giorgio Di Natale, Marie-Lise Flottes, and Bruno Rouzeyre. "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans". In: *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)*. IEEE. 2014, pp. 49–54.

[34] Niklas Eén and Niklas Sörensson. "An extensible SAT-solver". In: *International conference on theory and applications of satisfiability testing*. Springer. 2003, pp. 502–518.

[35] Hans Eisenmann and Frank M Johannes. "Generic global placement and floorplanning". In: *Proceedings of the 35th annual Design Automation Conference*. 1998, pp. 269–274.

[36] Mohammad Eslami, Johann Knechtel, Ozgur Sinanoglu, Ramesh Karri, and Samuel Pagliarini. "Benchmarking Advanced Security Closure of Physical Layouts: ISPD 2023 Contest". In: *Proceedings of the 2023 International Symposium on Physical Design*. 2023, pp. 256–264.

[37] Tiago Augusto Fontana, Erfan Aghaeekiasaraee, Renan Netto, Sheiny Fabre Almeida, Upma Gandh, Aysa Fakheri Tabrizi, David Westwick, Laleh Behjat, and José Luís Güntzel. "ILP-based global routing optimization with cell movements". In: *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE. 2021, pp. 25–30.

[38] Henri Fraisse and Dinesh Gaitonde. "A SAT-based timing driven place and route flow for critical soft IP". In: *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE. 2018, pp. 8–87.

[39] Stèphano MM Gonçalves, Leomar S da Rosa, and Felipe De S Marques. "A survey of path search algorithms for VLSI detailed routing". In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2017, pp. 1–4.

[40] Stèphano MM Gonçalves, Leomar S Rosa, and Felipe S Marques. "DRAPS: A design rule aware path search algorithm for detailed routing". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 67.7 (2019), pp. 1239–1243.

[41] Stèphano MM Gonçalves, Leomar S da Rosa Jr, and Felipe S Marques. "SmartDR: Algorithms and techniques for fast detailed routing with good design rule handling". In: *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 26.2 (2020), pp. 1–38.

[42] Guangxin Guo, Hailong You, Zhengguang Tang, Benzheng Li, Cong Li, and Xiaojue Zhang. "ASSURER: A PPA-friendly Security Closure Framework for Physical Design". In: *Proceedings of the 28th Asia and South Pacific Design Automation Conference*. 2023, pp. 504–509.

[43] Antonin Guttman. "R-trees: A dynamic index structure for spatial searching". In: *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*. 1984, pp. 47–57.

[44] Xu He, Wing-Kai Chow, and Evangeline FY Young. "SRP: Simultaneous routing and placement for congestion refinement". In: *Proceedings of the 2013 ACM International symposium on Physical Design*. 2013, pp. 108–113.

[45] Xu He, Tao Huang, Linfu Xiao, Haitong Tian, Guxin Cui, and Evangeline FY Young. "Ripple: An effective routability-driven placer by iterative cell movement". In: *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE. 2011, pp. 74–79.

[46] Xu He, Yao Wang, Yang Guo, and Evangeline FY Young. "Ripple 2.0: Improved movement of cells in routability-driven placement". In: *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 22.1 (2016), pp. 1–26.

[47] Zhuolun He, Yuzhe Ma, and Bei Yu. "X-Check: CPU-Accelerated Design Rule Checking via Parallel Sweepline Algorithms". In: *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 2022, pp. 1–9.

[48] Asmus Hetzel. "A sequential detailed router for huge grid graphs". In: *Proceedings Design, Automation and Test in Europe*. IEEE. 1998, pp. 332–338.

[49] Hamed Hossein-Talaee and Ali Jahanian. "Layout vulnerability reduction against trojan insertion using security-aware white space distribution". In: *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE. 2017, pp. 551–555.

[50] Jhih-Wei Hsu, Kuan-Cheng Chen, Yan-Syuan Chen, Yu-Hsiang Lo, and Yao-Wen Chang. "Security-aware Physical Design against Trojan Insertion, Frontside Probing, and Fault Injection Attacks". In: *Proceedings of the 2023 International Symposium on Physical Design*. 2023, pp. 220–228.

[51] Kai-Shun Hu, Ming-Jen Yang, Tao-Chun Yu, and Guan-Chuen Chen. "ICCAD-2020 CAD contest in routing with cell movement". In: *Proceedings of the 39th International Conference on Computer-Aided Design*. 2020, pp. 1–4.

[52] Kai-Shun Hu, Tao-Chun Yu, Ming-Jen Yang, and Chin-Fang Cindy Shen. "2021 ICCAD CAD Contest Problem B: Routing with Cell Movement Advanced". In: *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE. 2021, pp. 1–5.

[53] Chau-Chin Huang, Hsin-Ying Lee, Bo-Qiao Lin, Sheng-Wei Yang, Chin-Hao Chang, Szu-To Chen, Yao-Wen Chang, Tung-Chieh Chen, and Ismail Bustany. "NTUplace4dr: A detailed-routing-driven placer for mixed-size circuit designs with technology and region constraints". In: *IEEE Transactions*

on Computer-Aided Design of Integrated Circuits and Systems 37.3 (2017), pp. 669–681.

[54] Zhipeng Huang, Haishan Huang, Runming Shi, Xu Li, Xuan Zhang, Weijie Chen, Jiaxiang Wang, and Ziran Zhu. "Detailed placement and global routing co-optimization with complex constraints". In: *Electronics* 11.1 (2022), p. 51.

[55] Xiaotao Jia, Yici Cai, Qiang Zhou, Gang Chen, Zhuoyuan Li, and Zuowei Li. "MCFRoute: A detailed router based on multi-commodity flow method". In: *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE. 2014, pp. 397–404.

[56] Xiaotao Jia, Yici Cai, Qiang Zhou, and Bei Yu. "MCFRoute 2.0: A redundant via insertion enhanced concurrent detailed router". In: *Proceedings of the 26th edition on Great Lakes Symposium on VLSI*. 2016, pp. 87–92.

[57] Yun-Jhe Jiang and Shao-Yun Fang. "Pin Access-Oriented Concurrent Detailed Routing". In: *Proceedings of the 2023 International Symposium on Physical Design*. 2023, pp. 17–25.

[58] Andrew B Kahng, Jian Kuang, Wen-Hao Liu, and Bangqi Xu. "In-route pin access-driven placement refinement for improved detailed routing convergence". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41.3 (2021), pp. 784–788.

[59] Andrew B Kahng, Jens Lienig, Igor L Markov, and Jin Hu. *VLSI physical design: from graph partitioning to timing closure*. Vol. 312. Springer, 2011.

[60] Andrew B Kahng, Lutong Wang, and Bangqi Xu. "The tao of PAO: Anatomy of a pin access oracle for detailed routing". In: *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE. 2020, pp. 1–6.

[61] Andrew B Kahng, Lutong Wang, and Bangqi Xu. "TritonRoute-WXL: The Open-Source Router With Integrated DRC Engine". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41.4 (2021), pp. 1076–1089.

[62] Andrew B Kahng, Lutong Wang, and Bangqi Xu. "Tritonroute: An initial detailed router for advanced vlsi technologies". In: *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. ACM. 2018, pp. 1–8.

[63] Andrew B Kahng, Lutong Wang, and Bangqi Xu. "Tritonroute: The open-source detailed router". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40.3 (2020), pp. 547–559.

[64] Andrew B Kahng and Qinke Wang. "Implementation and extensibility of an analytic placer". In: *Proceedings of the 2004 international symposium on Physical design*. 2004, pp. 18–25.

[65] Rajit Karmakar and Santanu Chattopadhyay. "On securing scan obfuscation strategies against ScanSAT attack". In: *2020 21st International Symposium on Quality Electronic Design (ISQED)*. IEEE. 2020, pp. 213–218.

[66] Myung-Chul Kim, Jin Hu, Dong-Jin Lee, and Igor L Markov. "A SimPLR method for routability-driven placement". In: *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE. 2011, pp. 67–73.

[67] Myung-Chul Kim, Dong-Jin Lee, and Igor L Markov. "SimPL: An effective placement algorithm". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31.1 (2011), pp. 50–60.

[68] Johann Knechtel, Jayanth Gopinath, Mohammed Ashraf, Jitendra Bhandari, Ozgur Sinanoglu, and Ramesh Karri. "Benchmarking Security Closure of

Physical Layouts: ISPD 2022 Contest". In: *Proceedings of the 2022 International Symposium on Physical Design*. 2022, pp. 221–228.

[69] Johann Knechtel, Jayanth Gopinath, Jitendra Bhandari, Mohammed Ashraf, Hussam Amrouch, Shekhar Borkar, Sung-Kyu Lim, Ozgur Sinanoglu, and Ramesh Karri. "Security closure of physical layouts ICCAD special session paper". In: *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE. 2021, pp. 1–9.

[70] Yue-Sun Kuo, TC Chern, and Wei-Kuan Shih. "Fast algorithm for optimal layer assignment". In: *Integration* 7.3 (1989), pp. 231–245.

[71] Ulrich Lauther. "An O (N log N) algorithm for Boolean mask operations". In: *Papers on Twenty-five years of electronic design automation*. 1988, pp. 233–240.

[72] Chin Yang Lee. "An algorithm for path connections and its applications". In: *IRE transactions on electronic computers* 3 (1961), pp. 346–365.

[73] Haocheng Li, Gengjie Chen, Bentian Jiang, Jingsong Chen, and Evangeline FY Young. "Dr. CU 2.0: A scalable detailed routing framework with correct-by-construction design rule satisfaction". In: *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE. 2019, pp. 1–7.

[74] Shiju Lin, Jinwei Liu, Evangeline FY Young, and Martin DF Wong. "Gamer: Gpu accelerated maze routing". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2022).

[75] Shiju Lin and Martin DF Wong. "Superfast Full-Scale CPU-Accelerated Global Routing". In: *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 2022, pp. 1–8.

[76] Tao Lin, Chris Chu, Joseph R Shinnerl, Ismail Bustany, and Ivailo Nedelchev. "POLAR: Placement based on novel rough legalization and refinement". In: *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE. 2013, pp. 357–362.

[77] Yibo Lin, Shounak Dhar, Wuxi Li, Haoxing Ren, Brucek Khailany, and David Z Pan. "Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement". In: *Proceedings of the 56th Annual Design Automation Conference 2019*. 2019, pp. 1–6.

[78] Genggeng Liu, Zhen Zhuang, Wenzhong Guo, and Ting-Chi Wang. "RDTA: An efficient routability-driven track assignment algorithm". In: *Proceedings of the 2019 on Great Lakes Symposium on VLSI*. 2019, pp. 315–318.

[79] Jinwei Liu, Chak-Wa Pui, Fangzhou Wang, and Evangeline FY Young. "CUGR: Detailed-Routability-Driven 3D Global Routing with Probabilistic Resource Model". In: *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE. 2020, pp. 1–6.

[80] Lixin Liu, Bangqi Fu, Martin DF Wong, and Evangeline FY Young. "Xplace: an extremely fast and extensible global placement framework". In: *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 2022, pp. 1309–1314.

[81] Siting Liu, Yuan Pu, Peiyu Liao, Hongzhong Wu, Rui Zhang, Zhitang Chen, Wenlong Lv, Yibo Lin, and Bei Yu. "Fastgr: Global routing on cpu-gpu with heterogeneous task graph scheduler". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2022).

[82] Wen-Hao Liu, Wei-Chun Kao, Yih-Lang Li, and Kai-Yuan Chao. "NCTU-GR 2.0: Multithreaded collision-aware global routing with bounded-length maze

routing". In: *IEEE Transactions on computer-aided design of integrated circuits and systems* 32.5 (2013), pp. 709–722.

[83] Wen-Hao Liu, Cheng-Kok Koh, and Yih-Lang Li. "Optimization of placement solutions for routability". In: *Proceedings of the 50th Annual Design Automation Conference*. 2013, pp. 1–9.

[84] Wen-Hao Liu, Stefanus Mantik, Wing-Kai Chow, Yixiao Ding, Amin Farshidi, and Gracieli Posser. "ISPD 2019 initial detailed routing contest and benchmark with advanced routing rules". In: *Proceedings of the 2019 International Symposium on Physical Design*. 2019, pp. 147–151.

[85] Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, Dennis J- H Huang, Chin-Chi Teng, and Chung-Kuan Cheng. "ePlace: Electrostatics based placement using Nesterov's method". In: *Proceedings of the 51st Annual Design Automation Conference*. 2014, pp. 1–6.

[86] Jingwei Lu, Hao Zhuang, Pengwen Chen, Hongliang Chang, Chin-Chih Chang, Yiu-Chung Wong, Lu Sha, Dennis Huang, Yufeng Luo, Chin-Chi Teng, et al. "ePlace-MS: Electrostatics-based placement for mixed-size circuits". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34.5 (2015), pp. 685–698.

[87] Stefanus Mantik, Gracieli Posser, Wing-Kai Chow, Yixiao Ding, and Wen-Hao Liu. "ISPD 2018 initial detailed routing contest and benchmarks". In: *Proceedings of the 2018 International Symposium on Physical Design*. 2018, pp. 140–143.

[88] Andrea Marcelli, Marco Restifo, Ernesto Sanchez, and Giovanni Squillero. "An evolutionary approach to hardware encryption and trojan-horse mitiga-

tion". In: *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE. 2017, pp. 1593–1598.

[89] Michael D Moffitt. "MaizeRouter: Engineering an effective global router". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27.11 (2008), pp. 2017–2026.

[90] *mor1kx - an OpenRISC processor IP core*. https://github.com/openrisc/mor1kx.

[91] Alexander Nadel and Vadim Ryvchin. "Efficient SAT solving under assumptions". In: *International Conference on Theory and Applications of Satisfiability Testing*. Springer. 2012, pp. 242–255.

[92] Gi-Joon Nam, Cliff Sze, and Mehmet Yildiz. "The ISPD global routing benchmark suite". In: *Proceedings of the 2008 international symposium on Physical design*. 2008, pp. 156–159.

[93] Gi-Joon Nam, Mehmet Yildiz, David Z Pan, and Patrick H Madden. "ISPD placement contest updates and ISPD 2007 global routing contest". In: *Proceedings of the 2007 international symposium on Physical design*. 2007, pp. 167–167.

[94] *NanGate FreePDK45 Open Cell Library*. Nangate Inc. 2011. URL: http://www.nangate.com/?page_id=2325.

[95] William C Naylor, Ross Donelly, and Lu Sha. *Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer*. US Patent 6,301,693. 2001.

[96] Tim Nieberg. "Gridless pin access in detailed routing". In: *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE. 2011, pp. 170–175.

[97] Nils J Nilsson. “Problem-solving methods in”. In: *Artificial Intelligence* 5 (1971).

[98] Muhammet Mustafa Ozdal. “Detailed-routing algorithms for dense pin clusters in integrated circuits”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28.3 (2009), pp. 340–349.

[99] Min Pan and Chris Chu. “FastRoute 2.0: A high-quality and efficient global router”. In: *2007 Asia and south pacific design automation conference*. IEEE. 2007, pp. 250–255.

[100] Min Pan and Chris Chu. “FastRoute: A step to integrate global routing into placement”. In: *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*. 2006, pp. 464–471.

[101] Min Pan and Chris Chu. “IPR: An integrated placement and routing algorithm”. In: *Proceedings of the 44th Annual Design Automation Conference*. 2007, pp. 59–62.

[102] Min Pan, Natarajan Viswanathan, and Chris Chu. “An efficient and effective detailed placement algorithm”. In: *ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005.* IEEE. 2005, pp. 48–55.

[103] Jarrod A Roy, James F Lu, and Igor L Markov. “Seeing the forest and the trees: Steiner wirelength optimization in placemen”. In: *Proceedings of the 2006 international symposium on Physical design*. 2006, pp. 78–85.

[104] Jarrod A Roy and Igor L Markov. “High-performance routing at the nanometer scale”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27.6 (2008), pp. 1066–1077.

[105] Jarrod A Roy, Natarajan Viswanathan, Gi-Joon Nam, Charles J Alpert, and Igor L Markov. "CRISP: Congestion reduction by iterated spreading during placement". In: *Proceedings of the 2009 International Conference on Computer-Aided Design*. 2009, pp. 357–362.

[106] Mohammad Saleh Samimi, Ehsan Aerabi, Zahra Kazemi, Mahdi Fazeli, and Ahmad Patooghy. "Hardware enlightening: No where to hide your hardware trojans!" In: *2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE. 2016, pp. 251–256.

[107] M Sato, JB Kim, T Awashima, and T Ohtsuki. "A theoretically optimal and practically fast algorithm for VLSI geometrical design rule verification". In: *1988 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 1988, pp. 1445–1448.

[108] C-JR Shi. "Solving constrained via minimization by compact linear programming". In: *Proceedings of ASP-DAC'97: Asia and South Pacific Design Automation Conference*. IEEE. 1997, pp. 635–640.

[109] Daohang Shi and Azadeh Davoodi. "TraPL: Track planning of local congestion for global routing". In: *Proceedings of the 54th Annual Design Automation Conference 2017*. 2017, pp. 1–6.

[110] Dominik Sisejkovic, Farhad Merchant, Lennart M Reimann, and Rainer Leupers. "Deceptive logic locking for hardware integrity protection against machine learning attacks". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41.6 (2021), pp. 1716–1729.

[111] Dominik Šišejković, Farhad Merchant, Rainer Leupers, Gerd Ascheid, and Sascha Kegreiss. "Control-lock: Securing processor cores against software-

controlled hardware trojans". In: *Proceedings of the 2019 on Great Lakes Symposium on VLSI*. 2019, pp. 27–32.

[112] Sergei Skorobogatov and Christopher Woods. "Breakthrough silicon scanning discovers backdoor in military chip". In: *Cryptographic Hardware and Embedded Systems–CHES 2012: 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings 14*. Springer. 2012, pp. 23–40.

[113] JIRI Soukup. "Fast maze router". In: *Design Automation Conference*. IEEE Computer Society. 1978, pp. 100–101.

[114] Peter Spindler, Ulf Schlichtmann, and Frank M Johannes. "Abacus: Fast legalization of standard cell circuits with minimal movement". In: *Proceedings of the 2008 international symposium on Physical design*. 2008, pp. 47–53.

[115] Peter Spindler, Ulf Schlichtmann, and Frank M Johannes. "Kraftwerk2—A fast force-directed quadratic placement approach using an accurate net model". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27.8 (2008), pp. 1398–1411.

[116] Fan-Keng Sun, Hao Chen, Ching-Yu Chen, Chen-Hao Hsu, and Yao-Wen Chang. "A multithreaded initial detailed routing algorithm considering global routing guides". In: *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE. 2018, pp. 1–7.

[117] Timothy Trippel, Kang G Shin, Kevin B Bush, and Matthew Hicks. "ICAS: an extensible framework for estimating the susceptibility of ic layouts to additive trojans". In: *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2020, pp. 1742–1759.

[118] Natarajan Viswanathan, Min Pan, and Chris Chu. "FastPlace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control". In: *2007 Asia and South Pacific Design Automation Conference*. IEEE. 2007, pp. 135–140.

[119] 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)*. IEEE. 2016, pp. 378–383.

[120] Tai-Hsuan Wu, Azadeh Davoodi, and Jeffrey T Linderoth. "GRIP: Scalable 3D global routing using integer programming". In: *Proceedings of the 46th Annual Design Automation Conference*. 2009, pp. 320–325.

[121] Kan Xiao, Domenic Forte, Yier Jin, Ramesh Karri, Swarup Bhunia, and Mohammad Tehranipoor. "Hardware trojans: Lessons learned after one decade of research". In: *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 22.1 (2016), pp. 1–23.

[122] Kan Xiao, Domenic Forte, and Mohammed Tehranipoor. "A novel built-in self-authentication technique to prevent inserting hardware trojans". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33.12 (2014), pp. 1778–1791.

[123] Xiaoqing Xu, Brian Cline, Greg Yeric, Bei Yu, and David Z Pan. "Self-Aligned Double Patterning Aware Pin Access and Standard Cell Layout Co-Optimization". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34.5 (2015), pp. 699–712.

[124] Xiaoqing Xu, Yibo Lin, Vinicius Livramento, and David Z Pan. "Concurrent pin access optimization for unidirectional routing". In: *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE. 2017, pp. 1–6.

[125] Xiaoqing Xu, Bei Yu, Jhih-Rong Gao, Che-Lun Hsu, and David Z Pan. "PARR: Pin-access planning and regular routing for self-aligned double patterning". In: *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 21.3 (2016), pp. 1–21.

[126] Yue Xu and Chris Chu. "MGR: Multi-level global router". In: *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE. 2011, pp. 250–255.

[127] Yue Xu, Yanheng Zhang, and Chris Chu. "FastRoute 4.0: Global router with efficient via minimization". In: *2009 Asia and South Pacific Design Automation Conference*. IEEE. 2009, pp. 576–581.

[128] Kaiyuan Yang, Matthew Hicks, Qing Dong, Todd Austin, and Dennis Sylvester. "A2: Analog malicious hardware". In: *2016 IEEE symposium on security and privacy (SP)*. IEEE. 2016, pp. 18–37.

[129] Xinshi Zang, Fangzhou Wang, Jinwei Liu, and Martin DF Wong. "ATLAS: A Two-Level Layer-Aware Scheme for Routing with Cell Movement". In: *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 2022, pp. 1–7.

[130] Yanheng Zhang and Chris Chu. "CROP: Fast and effective congestion refinement of placement". In: *Proceedings of the 2009 International Conference on Computer-Aided Design*. 2009, pp. 344–350.

[131] Yanheng Zhang and Chris Chu. "RegularRoute: An efficient detailed router applying regular routing patterns". In: *IEEE transactions on very large scale integration (VLSI) systems* 21.9 (2012), pp. 1655–1668.

[132] Ziran Zhu, Fuheng Shen, Yangjie Mei, Zhipeng Huang, Jianli Chen, and Jun Yang. "A Robust Global Routing Engine with High-Accuracy Cell Movement under Advanced Constraints". In: *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 2022, pp. 1–9.

[133] Zhen Zhuang, Genggeng Liu, Tsung-Yi Ho, Bei Yu, and Wenzhong Guo. "TRADER: a practical track-assignment-based detailed router". In: *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2022, pp. 766–771.

[134] Peng Zou, Zhijie Cai, Zhifeng Lin, Chenyue Ma, Jun Yu, and Jianli Chen. "Incremental 3D Global Routing Considering Cell Movement and Complex Routing Constraints". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2022).

[135] Peng Zou, Zhifeng Lin, Chenyue Ma, Jun Yu, and Jianli Chen. "Late Breaking Results: Incremental 3D global routing considering cell movement". In: *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE. 2021, pp. 1366–1367.

List of Publications

[1] Wei Li, Fangzhou Wang, José MF Moura, and RD Shawn Blanton. "Global Floorplanning via Semidefinite Programming". In: *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE. 2023, pp. 1–6.

[2] Shixiong Kai, Chak-Wa Pui, Fangzhou Wang, Shougao Jiang, Bin Wang, Yu Huang, and Jianye Hao. "TOFU: A Two-Step Floorplan Refinement Framework for Whitespace Reduction". In: *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2023, pp. 1–5.

[3] Fangzhou Wang, Jinwei Liu, and Evangeline FY Young. "FastPass: Fast Pin Access Analysis with Incremental SAT Solving". In: *Proceedings of the 2023 International Symposium on Physical Design*. 2023, pp. 9–16.

[4] Fangzhou Wang, Qijing Wang, Bangqi Fu, Shui Jiang, Xiaopeng Zhang, Lilas Alrahis, Ozgur Sinanoglu, Johann Knechtel, Tsung-Yi Ho, and Evangeline FY Young. "Security Closure of IC Layouts Against Hardware Trojans". In: *Proceedings of the 2023 International Symposium on Physical Design*. 2023, pp. 229–237.

[5] Xinshi Zang, Fangzhou Wang, Jinwei Liu, and Martin DF Wong. "ATLAS: A Two-Level Layer-Aware Scheme for Routing with Cell Movement". In:

Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design. 2022, pp. 1–7.

[6] Fangzhou Wang, Lixin Liu, Jingsong Chen, Jinwei Liu, Xinshi Zang, and Martin DF Wong. "Starfish: An Efficient P&R Co-Optimization Engine with A*-based Partial Rerouting". In: *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE. 2021, pp. 1–9.

[7] Bentian Jiang, Jingsong Chen, Jinwei Liu, Lixin Liu, Fangzhou Wang, Xiaopeng Zhang, and Evangeline FY Young. "CU. POKer: Placing DNNs on WSE With Optimal Kernel Sizing and Efficient Protocol Optimization". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41.6 (2021), pp. 1888–1901.

[8] Bentian Jiang, Jingsong Chen, Jinwei Liu, Lixin Liu, Fangzhou Wang, Xiaopeng Zhang, and Evangeline FY Young. "CU. POKer: placing DNNs on wafer-scale AI accelerator with optimal kernel sizing". In: *Proceedings of the 39th International Conference on Computer-Aided Design*. 2020, pp. 1–9.

[9] Jinwei Liu, Chak-Wa Pui, Fangzhou Wang, and Evangeline FY Young. "CUGR: Detailed-Routability-Driven 3D Global Routing with Probabilistic Resource Model". In: *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE. 2020, pp. 1–6.