Advanced Node Physical Design, Attack and Anti-attack

LI, Haocheng

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
July 2020

Thesis Assessment Committee

Professor YU Bei (Chair)

Professor YOUNG Fung Yu (Thesis Supervisor)

Professor XU Qiang (Committee Member)

Professor HU Jiang (External Examiner)

Abstract

of thesis entitled:

Advanced Node Physical Design, Attack and Anti-attack

submitted by LI, Haocheng

for the degree of Doctor of Philosophy

at The Chinese University of Hong Kong in July 2020

The advancement of semiconductor technology shrinks the feature size but boosts the number of design rules and requires globalized manufacturing. Crucial problems have emerged in integrated circuit physical design where placement and routing are two critical stages. Provided by high-end manufacturers, multi-row-high standard cells resolve the lack of intra-cell routability but bring challenges to traditional placers on efficiency and effectiveness. Due to the increasing density of cell pins and power mesh, interference between pins and mesh limits the number of access points for each pin.

This thesis considers pin access and complicated design rules in both detailed placement and detailed routing stages. In placement, we propose a legalization method for mixed-cell-height circuits by a window-based cell insertion technique. Constraints on fence region, edge spacing, and pin access are considered, besides providing an overlap-free solution close to the global placement solution and fulfilling the power mesh alignment. For a legalized placement, we propose two network-flow-based

methods jointly optimizing the maximum and average displacement. In routing, our proposed detailed router judiciously accesses pins without valid on-grid access points and handles new design rules such as length-dependent parallel-run-length spacing, end-of-line spacing with parallel edges, and corner-to-corner spacing.

With studies on physical design in advanced technology nodes, this thesis analyzes the security of physical design solutions under split manufacturing and proposes methods to protect layouts from potential malice. The notion of the integrated circuit split manufacturing that delegates the front-end-of-line (FEOL) and back-end-of-line (BEOL) parts to two foundries, is to prevent attack by adversaries in the FEOL facility. Traditional security-oblivious physical design tools place connected components close to each other, which discloses the BEOL connections. We challenge the security promise of split manufacturing by formulating various layout-level placement and routing hints as vector-based and image-based features and construct a sophisticated deep neural network that can infer the missing BEOL connections with high accuracy. Our proposed SoftMax regression loss allows our attack to directly and effectively select the most probable BEOL connection among relevant candidates. To escalate the security level of layouts, we further propose a random blockage insertion strategy that can be smoothly integrated into any commercial physical design flow.

摘要

先進製程物理設計、攻擊與反攻擊

半導體製程的演進在減小器件特徵尺寸同時，要求更複雜的設計規則和更加深度全球化的製造生態，並為積體電路物理設計帶來了新的挑戰。其中，佈局與布線是對版圖有決定性作用的兩個重要階段。為解決標準單元庫中單行單元缺乏單元內可佈線性的問題，高端製造廠引入多行單元以實現較為複雜或驅動力較大的標準單元，使傳統佈局器難以勝任混合高度單元佈局。由於單元引腳和電源網格密度的增加，引腳與引腳、引腳與電源之間的干涉使得引腳接入點的選擇受到了很大的限制。本文在詳細佈局和詳細佈線中考慮引腳可接入性及其他複雜設計規則，提出了一種基於單元插入的混合高度單元合法化演算法，利用最小的單元移動，在消除單元重合的同時，滿足電源對齊、圍欄限制、邊緣間距、引腳可接入性等約束。

另一方面，無晶圓設計公司依賴非受控的離岸代工廠以實現先進製程的廉價導入，致使智慧財產權暴露於多種攻擊手段。為避免掌握先進製程的離岸代工廠侵犯智慧財產權，分離製造旨在僅將積體電路的基板交付離岸代工廠製造，並由內部整合或其他可信賴的部門完成配線工程，以期提高智慧財產權及相關設備與基礎設施的安全性。然而，非設防的傳統物理設計工具會將配線相連的標準單元擺放在相近的位置，進而造成智慧財產權洩漏。

基於對先進製程物理設計的研究，本文對物理設計結果進行安全性評估並提出反攻擊方法以保護版圖。我們將基板中的佈局佈線資訊轉化為向量或張量形式的特徵，並構建了一種精巧的深度神經網路以有效預測基板中的網表片段在配線中的連

接方式。我們提出了歸一化回歸損失函數，以直接為每個負載選出最可能的驅動。我們進一步提出一種易於集成到任何商業物理設計流程的隨機障礙策略，以提高版圖安全性。

摘要

先进制程物理设计、攻击与反攻击

半导体制程的演进在减小器件特征尺寸同时，要求更复杂的设计规则和更加深度全球化的制造生态，并为集成电路物理设计带来了新的挑战。其中，布局与布线是对版图有决定性作用的两个重要阶段。为解决标准单元库中单行单元缺乏单元内可布线性的问题，高端制造厂引入多行单元以实现较为复杂或驱动力较大的标准单元，使传统布局器难以胜任混合高度单元布局。由于单元引脚和电源网格密度的增加，引脚与引脚、引脚与电源之间的干涉使得引脚接入点的选择受到了很大的限制。本文在详细布局和详细布线中考虑引脚可接入性及其他复杂设计规则，提出了一种基于单元插入的混合高度单元合法化算法，利用最小的单元移动，在消除单元重合的同时，满足电源对齐、围栏限制、边缘间距、引脚可接入性等约束。

另一方面，无晶圆设计公司依赖非受控的离岸代工厂以实现先进制程的廉价导入，致使知识产权暴露于多种攻击手段。为避免掌握先进制程的离岸代工厂侵犯知识产权，分离制造旨在仅将集成电路的基板交付离岸代工厂制造，并由内部整合或其他可信赖的部门完成配线工程，以期提高知识产权及相关设备与基础设施的安全性。然而，非设防的传统物理设计工具会将配线相连的标准单元摆放在相近的位置，进而造成知识产权泄漏。

基于对先进制程物理设计的研究，本文对物理设计结果进行安全性评估并提出反攻击方法以保护版图。我们将基板中的布局布线信息转化为向量或张量形式的特征，并构建了一种精巧的深度神经网络以有效预测基板中的网表片段在配线中的连

接方式。我们提出了归一化回归损失函数，以直接为每个负载选出最可能的驱动。我们进一步提出一种易于集成到任何商业物理设计流程的随机障碍策略，以提高版图安全性。

In memory of COVID-19 victims and tribute to all first responders and front-line workers.

Acknowledgement

First of all, I would like to express my deepest gratitude and appreciation to my advisor Professor Evangeline F.Y. Young who leads me to the area of logic synthesis and physical design. This thesis would be impossible without her guidance, advice, support, and sharing of extensive experience, profound knowledge, and firm conviction.

Besides my adviser, I also want to express my thanks to the rest of my doctoral committee, Professor Bei Yu, Professor Qiang Xu, and Professor Jiang Hu, for their insightful comments and constructive suggestions. In particular, I very much appreciate Professor Bei Yu for various collaborations on research projects and helpful discussions on the sense of aesthetics. I wish to express my sincere thanks to Professor Qiang Xu for his valuable comments on hardware security and machine learning. I am very grateful to Professor Jiang Hu for his generous support on detailed routing and split manufacturing.

I would also like to extend my gratitude to my internship mentors at Cadence: Dr. Wing-Kai Chow, Dr. Zhuo Li, and Dr. Mehmet C. Yildiz, for their constructive suggestions on career aspiration and personality development. I am also very thankful to industrial colleagues in early global route team: Dr. Yixiao Ding, Dr. Derong Liu, and Dr. Gracieli Posser, for their active discussions on electronic design automation algorithms and enterprise-level software development, and other colleagues in Austin, Texas: Dr. Charles J. Alpert, Dr. Amin Farshidi, Dr. Jhih-Rong Gao, Dr. Sheng-En Lin, Dr. Wen-Hao Liu, Dr. Natarajan Viswanathan, Brian Wilson, and Dr. Boyu Zhang for

their continuous help through and after my internship.

My sincere thanks also go to Dr. Jian Kuang for his share of foresight and sagacity on this industry at the Engineering Workshop, and other alumni and colleagues at The Chinese University of Hong Kong in Ma Liu Shui: Dr. Peishan Tu, Dr. Gengjie Chen, Dr. Chak-Wa Pui, Ka-Chun Lam, Hang Zhang, Chong-Wing Cheung, Jingsong Chen, Bentian Jiang, Jinwei Liu, Xiaopeng Zhang, Dan Zheng, Lixin Liu, Fangzhou Wang. It was my great honor to work with them. I also appreciate very much for the help of Dr. Yuzhe Ma, Dr. Huan Shi, Dr. Haoyu Yang, Tinghuan Chen, Hao Geng, Ran Chen, Qi Sun, and Lu Zhang.

I would also like to recognize the help and effort from Professor Ozgur Sinanoglu, Dr. Johann Knechtel, Mohammed Ashraf, Satwik Patnaik, and Abhrajit Sengupta, for the collaboration on split manufacturing, Professor Jianli Chen for helpful discussions and binary sharing, and Yueqing Zhang and ShuTing Cheng for the review and comments on the abstract in simplified and traditional Chinese.

Besides, I am very grateful to Enlightened communities in Shaanxi, Hong Kong, and Shanghai: Joey Bai, Yunxuan Bu, @Davidia, Yuefeng Mo, @prpr9, @Sherry33, @tijiao, Xiaotian Ye, Kang Zhang, and @ZhangBH. They were illuminators during my exploration of corners and end-of-lines in metropolitan and rural areas.

Finally, this thesis would not exist without the support and sacrifice of my family. I want to express my deepest gratitude to my parents Weimin Li and Fang Zhao for their unconditional love and to my cousins: Yuechen Tao, Hanlin Li, Mingwei Li, and Yunfan Huang, for their encouragement and support.

Contents

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

Summary

This chapter introduces the background of the very-large-scale integration (VLSI), the standard-cell-based methodology that effectively divides the design flow into stages, and the design rules that assure the quality of results (QoR) in each stage. It also sketches the hardware security problem raised under the context of contemporary globalization.

1.1 Background

The entire architecture of modern digital world is constructed upon an immense volume of software which is executed on integrated circuits (ICs). From the 2048 KHz Apollo Guidance Computer (AGC) with thousands of silicon-integrated three-input NOR gates and 2048-word magnetic-core memory (O'Brien, 2010) to the processors, storage, and peripherals collaborating to transfer, store, and display this thesis, the semiconductor feature size shrinks, the switching power reduces, the computing fre-

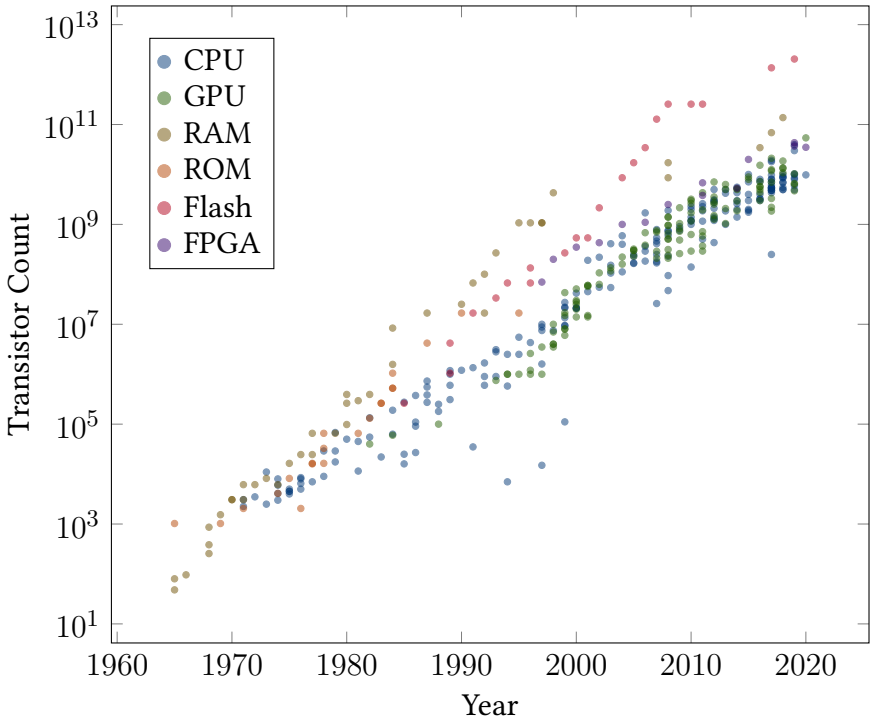


Figure 1.1: Transistor count on ICs. Data source: WikiChip and Wikipedia.

quency boosts, and the integration density increases. Moore's Law, which predicted a doubling of transistor count every two years due to the downsizing of device structures (Moore et al., 1975), was faithfully answered by the development of semiconductor technology nodes and automated design methodology. In the past five decades, generations and generations of central processing units (CPUs), graphics processing units (GPUs), and more recently field-programmable gate-arrays (FPGAs) are launched with exponentially growing numbers of transistors, which approximates a straight line when plotted on semilog Fig. 1.1. Although the 40% increase per year requires an annual 15% increase in investment, the cost per transistor decreases by 25% and the number of units sold also increases by 15% each year (Wong, 2003). With the globalized and diversified design and fabrication of ICs, the feature size of microscopic devices continuously narrows down to three nanometers and the gate height shrinks to three tracks (Sherazi et al., 2019) while macroscopic devices in our pockets,

on our wrists, and everywhere else become more powerful and ubiquitous, and will gradually become a part of tacit knowledge of human beings (Polanyi, 2009).

As early as ICs began to incorporate hundreds of gates and thousands of transistors, the computers they enabled were harnessed to speed the development process and eliminate design errors (Laws, 2010). The modern flow of very-large-scale integration (VLSI) design is separated into distinct stages including logic synthesis, physical design, physical verification, etc. (Kahng et al., 2011), which is made possible by the standard-cell-based methodology that isolates both the logical and physical communities from the challenges faced by each other. While physical design includes other steps such as floorplanning, power mesh routing, and clock tree synthesis, this thesis will focus on placement and routing, and their effect to circuit efficiency and security.

As the name suggested, the standard cell placement stage of physical synthesis allocates standard cells on sites of the core area. Traditionally, placement consists of three steps: global placement, legalization, and detailed placement. Global placement imports cells from the netlist and spreads the cells optimizing wirelength with analytical models (Gessler et al., 2020). Multiple objectives are considered including timing, congestion, and power but physical constraints like overlapping, site alignment, and special net connection are neglected. The physical constraints are instead resolved in the later step, legalization, where the quality of global placement solution is preserved as much as possible, i.e., the average displacement of cells is minimized (Darav et al., 2017). Finally, refinements are conducted in detailed placement where the wirelength is further reduced while preserving physical constraints (Khasawneh & Madden, 2020).

To handle complicated design rules and large solution space, regular net routing is typically divided into global routing and detailed routing. Global routing partitions the whole routing region into bi-dimensional (2D) or tri-dimensional (3D) global rout-

ing cells (G-cells) and constructs edges representing multiple tracks (or multiple vias) to estimate congestion and timing (J. Liu et al., 2020). For each net, global routing provides a collection of G-cells as output and assumes the final path will be close to the shortest path within the G-cells (Dolgov et al., 2019). The detailed routing phase, which generates exact rectilinear wiring interconnects satisfying all design rules, has no success guaranty even with days of turnaround time and thus designer-intervened engineering change order (ECO) may be required (Park et al., 2019). Unlike global routing, detailed routing needs to handle a large number of design rules on a huge and detailed routing grid graph.

The effort of semiconductor technology and automation methodology does not wax and wane against each other like the shadowed part and the lightened part of the moon. Instead, with the advancement of technology nodes, new crucial problems are brought to automated physical design like having complicated rule checking, heterogeneous standard cells, and globalized manufacturing. The remaining of this chapter will detail the challenges from these three aspects. Section 1.2 will explain the nature of design rules to guarantee layout manufacturability. Section 1.3 will introduce the evolving cell library forced by the diminished feature size. Section 1.4 will investigate the potential malice under offshore manufacturing of fabless design houses. The overview of this thesis will be listed in Sec. 1.5.

1.2 Design Rules

While the feature size scales faster than the optical lithography wavelength, physical design becomes even more complex and circumscribed for the guarantee of manufacturable layouts and has thus become an even harder problem to resolve in the VLSI flow, not only because of the increased size and scale, but also due to the introduction of tedious design rules.

Obviously, two metal shapes from distinct nets are not allowed to overlap. Otherwise, these two nets will be short. More strictly, a spacing is required between any two metal shapes to guarantee that they are properly separated after lithography. Usually, the minimum spacing is at least the same as the width of regular wires in the layer. For special nets (i.e., power meshes) that are wider than regular wires, a wider spacing is required. Capacitive crosstalk between nearby signals becomes severe in sub-90 nm technologies and high density of wires can lead to an increase in capacitance due to coupling effect (X. Zhang et al., 2020). Among other repairing solutions including shielding and wire reordering, spacing increasing is one of the most common and direct solutions. Therefore, the parallel run length (PRL) spacing is introduced as a distance requirement that monotonously increases regarding to both the width and the PRL of the two metal shapes (Qi et al., 2015).

The Euclidean parallel run length (PRL) spacing generally applies on every direction around the metal: at the length side, width side, and corners. However, a special care is needed at line ends which are edges between two convex vertices closer than some threshold to each other. Sub-wavelength lithography without optical proximity correction (OPC) creates remarkable mismatches between mark layouts and wafer images such as narrower line ends. These insufficient connections with vias (i.e., vertical interconnect accesses) are hence more likely to be affected by electromigration and the irregularities induce aggravate variations on timing and power (S. Hu & Hu, 2007). Therefore, the design rule requires extra spacing near end-of-lines (EOLs) to reserve space for layout enhancement. In addition, metal EOL with parallel edges causes extra stitches in double patterning lithography, which may lead to yield degradation or even cause native conflicts (NCs) that cannot be resolved by any kind of stitch insertion (W. Li et al., 2020). A new kind of EOL spacing requirement that depends on the existence of other parallel edges is defined for layout generation with lithography-friendliness.

PRL spacing, EOL spacing, and other design rules restrict the solution space of physical design and increase the complexity for optimization. While the number of design rules provided by foundries was only a few dozen at 180 nm node, it increased for three times from 130 nm to 90 nm nodes to avoid manufacturing-unfriendly patterns and to achieve economically-acceptable yield level (Cote et al., 2004). The number of manufacturability-driven rules increased to several hundreds at 65 nm node and is still rising as technology node advances (Pan et al., 2010).

1.3 Standard Cells

Standard cells (or ploycells) are designed with the same height and aligned on placement sites for the abstraction of physical synthesis. Historically, standard cells were arranged in vertically separated rows with pads placed around the periphery of the chip (Sechen & Sangiovanni-Vincentelli, 1986). Between rows were two-layer wiring channels occupying a high fraction of die area and channel routing was conducted within channels to connect standard cells across the banks (Yoshimura & Kuh, 1982). Later, with the introduction of large macros, the rip-up-and-reroute scheme and two-layer 2D routers were proposed to tackle regions with irregular boundaries and non-convex shapes (Shin & Sangiovanni-Vincentelli, 1987). By elevating wires to higher metal layers, the standard cell rows were shifted abutting each other for the efficiency of die area. Consequently, the increasing of routing layers and the diversifying of wire width and timing characteristics transformed over-the-cell routing to a complicated 3D problem.

With the scaling down of feature size, the track height reduction of standard cells becomes another mandatory technique to save area. Internal routability within a single-site-high cell becomes inadequate due to the significant diminishing of in-cell tracks. Figure 1.2 illustrates the wire width (i.e., the shorter dimension of a

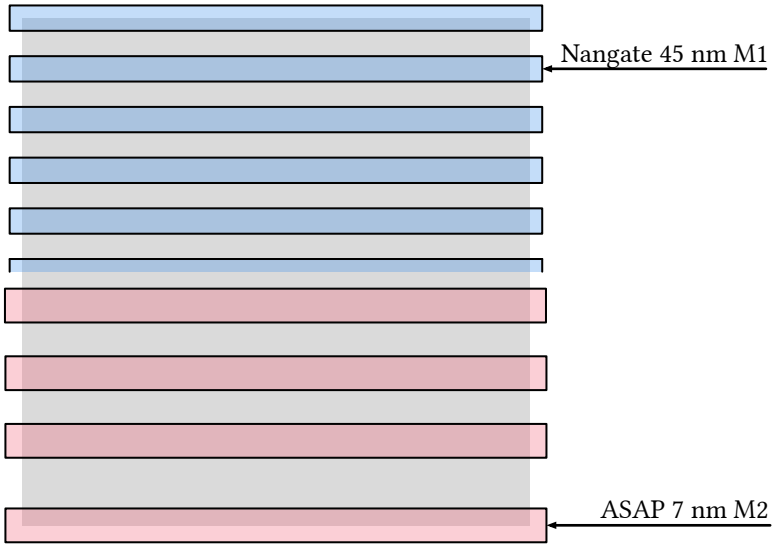


Figure 1.2: Wire width in each horizontal layer scaled to the same row height.

metal shape) in each layer with horizontal preferred routing direction of two cell libraries scaled to the same row height (gray). In the blue-colored Nangate 45 nm open cell library, standard cells are ten-track-high (Knudsen, 2008) while the height of standard cells in red-colored ASAP 7 nm cell library becomes seven tracks and a half (Vashishtha & Clark, 2019). The strengthening limitation of row height results in intra-cell routing congestion for complex standard cells like muxes (i.e., multiplexers, Baek et al. (2008)) and multi-bit flip-flops (Santos et al., 2012). In sub-5 nm technology nodes, high driving strength cells including AOI and XOR are also designed with multi-row height (Sherazi et al., 2019). On the other hand, cell area is wasted in the region for N-type transistors when large cell width is required by P-type transistors to achieve similar rise and fall transition time (Yu et al., 2002). For ascending performance and efficiency, complex cells are now designed with multi-row height and two-layer metal routing while simple cells remain single-row height and one-layer metal routing (Sherlekar, 2014). Consequently, the challenges in physical synthesis migrate from cell design to cell placement. As long as the physical constraints (e.g., cell overlapping) are not considered in global placement, the introduction of mix-cell-

height circuits transfers most of the problems to the legalization step.

Another consequence after track height reduction is that pin access to standard cells (Frederick, 2014) and lower-layer metal routing (H. Li, Patnaik, et al., 2019) become more difficult and confined due to increased pin density, limited access points, and interference between pins. In some cell libraries, there are some pins embraced by other pins and intra-cell wires so there is no violation-free grid point for these pins in Metal 1 (W.-H. Liu et al., 2019), which requires boundary-breaking methods for pin access in detailed routing.

1.4 Hardware Security

The introduction of cell library not only abstracts the synthesis flow into logic and physical phases but also passes a set of manufacturing standard from the foundry to design houses. While becoming a integrated device manufacturer (IDM) or owning a trustworthy foundry is not economically and technically viable, fabless design houses rely on offshore foundries for cost-effective access to the state-of-the-art technology nodes. This globalized and diversified nature of the IC supply chain including design, synthesis, fabrication, and distribution makes hardware as vulnerable as software. With the complete knowledge of exposed layouts, various attack avenues on intellectual property (IP) are enabled in the external foundries and the critical infrastructure deploying these ICs can be disrupted (Perez & Pagliarini, 2020). Malicious suppliers are able to steal the designs and underlying IP without effective protection from conventional passive methods, e.g., patents and copyrights (Rahman et al., 2020). Attackers may counterfeit chips by scanning all the layers (G. L. Zhang et al., 2020) or even insert hardware Trojans (X. Hu et al., 2020). The Organisation for Economic Co-operation and Development (OECD) estimated 138 billion US dollar of global fake export on electrical machinery and electronics in 2016 (OECD & EUIPO, 2019), mak-

ing it a category with the highest value of fake trade in the Harmonized Commodity Description and Coding Systems (HS).

Observe that the fabrication of higher metal layers with the same technology node as the transistor layer and lower metal layers is dispensable. For example, the wire width for Metal 9 is more than ten times that of Metal 3 in Nangate 45 nm cell library to provide better performance characteristics. By the 32 nm node, wires in top layers can be twenty times thicker than wires in the bottom layers (Alpert et al., 2010). While approving the feasibility of separately manufacturing the front-end-of-line (FEOL, i.e., the device layer and a few lower metal layers) and back-end-of-line (BEOL, i.e., the higher metal layers) parts of design (B. Hill et al., 2013), the Intelligence Advanced Research Projects Activity (IARPA) agency advocated split manufacturing to safeguard chip designs from potentially malicious foundries (McCants, 2016). A high-end but untrusted foundry fabricates the device layer and a few lower metal layers, whereas a low-end but trusted facility, which is possibly in-house, integrates the BEOL on top of the FEOL. When doing so, the untrusted foundries cannot get control of the full design while the fabless design houses can still avoid the wanton hemorrhage of both time and coin on the latest technology node and Vaidyanathan et al. (2014) verified this methodology showing no noticeable impact on circuit performance. However, merely splitting the designs into FEOL and BEOL may fall short in terms of security, pressing research regarding the attack and defense of split manufacturing.

1.5 Organization

The remainder of the thesis is organized as follows. In Chapter 2, we will review the previous works on detailed placement, detailed routing, and split manufacturing. In Chapter 3, under the circumstances of mixed-height standard cells, we demonstrate a multi-threading legalization optimizing the displacement from global placement po-

sitions. In Chapter 4, we demonstrate two network-flow-based detailed placement algorithms optimizing maximum and average displacement. In Chapter 5, we propose a detailed routing framework handling pin access and other design rules. In Chapter 6, we describe a deep-learning-based split manufacturing attack with novel feature extraction, network architecture, and lose function. In Chapter 7, we describe a split manufacturing defense by routing perturbation, followed by a conclusion in Chapter 8.

□ End of chapter.

Chapter 2

Literature Review

Summary

This chapter introduces related works in the area of physical design, attack, and anti-attack. For detailed placement, single-height methods, fixed-order optimization, and window-based incremental placement are discussed. For detailed routing, algorithms investigating pin access, path search, and different design rules are enumerated. The discussion on split manufacturing is divided into attack and defense parts.

2.1 Detailed Placement

The earliest literature assumes all multi-row cell as fixed macros and only conduct single-row cell legalization. For example, Tetris sorts cells by x-coordinate (then by width in case the x-coordinate is the same) and assigns the nearest available site for each cell (D. Hill, 2002). Abacus conducts a more sophisticated legalization algorithm to optimize the quadratic displacement in each row by dynamic program-

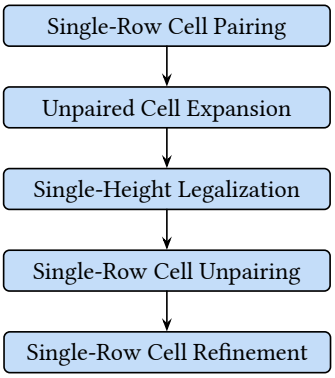


Figure 2.1: Legalization flow of (Wu & Chu, 2016).

ming (Spindler et al., 2008). Increasing number of multi-row cells causes large degradation in these methods and urges for dedicated mixed-cell-height algorithms. Later works convert double-row-cell legalization into a single-cell-height problem. Following that, other previous works on legalization and detailed placement algorithms for mixed-cell-height circuits can be categorized into three types: fixed-order (and fix-row) placement, window-based placement, and placement with other constraints.

2.1.1 Single-height Placement

Wu and Chu (2016) constructed artificial double-row cells from pairs of single-row cells by graph matching and conduct a double-row detailed placement on cells with the same double-row height. The proposed flow is shown in Fig. 2.1. When constructing the matching graph, they considered width difference, cell connectivity, and displacement in the edge weight of the graph. To solve the maximum weighted matching problem, a $\frac{1}{2}$-approximate path growing algorithm was used to achieve nearly optimal solution in linear running time (Drake & Hougardy, 2003). Details of the path growing algorithm is shown in Algorithm 2.1. This method is under the assumption that all double-row-high cells have the same power and ground rail configuration so they can only be placed on even rows.

Algorithm 2.1 Path Growing (Drake & Hougardy, 2003)

Require: Graph $G = (\mathcal{V}, \mathcal{E})$, weight $w\colon \mathcal{E} \to \mathbb{R}_+$.
Ensure: $\frac{1}{2}$-approximate maximum weight.

```
M_0 := ∅;
M_1 := ∅;
i := 0;
while E ≠ ∅ do
    Arbitrarily choose x ∈ V of degree at least 1;
    while x has a neighbor do
        Let {x, y} be the heaviest edge incident to x;
        M_i := M_i ∪ {{x, y}};
        i := 1 − i;
        Remove x from G;
        x := y;
    end while
end while
return max(w(M_0), w(M_1));
```

Dobre et al. (2017) partitioned the die area and assigned a specific cell height for each part. To begin with, the modified cell library is constructed containing all cell types with the same height. For each cell type t_i in the original cell library whose height and width are h_i and w_i, a modified cell type $\tilde{t}_i$ with height h_0 and width $\lceil \frac{h_i w_i}{h_0} \rceil$ is included in the modified cell library, where h_0 is the shortest cell height in the original library. Using single-height legalization algorithms, an initial solution is conducted with this modified cell library. The die area is then partitioned and a particular cell height was then specified to each partition based on the initial placement and the distribution of original cell height. The cells in each partition are sized to the specified height and the conventional algorithms are performed again within partition for the final solution. Its limitation is that all cell functions need to have implementations with every cell height. Empirical results show that mixed-cell-height designs achieve area and power reduction compared with single-cell-height designs but cell height swapping is required in this procedure and, as a strong and unnecessary constraint, selecting the same cell height for each floorplan partition affects the

estimation of timing and power in logic synthesis and highly restricts the solution space.

2.1.2 Fixed-order Placement

Similarly to the single-cell-height circuit legalizer Tetris (D. Hill, 2002) and dynamic-programming-based Abacus (Spindler et al., 2008), the second type of mixed-cell-height circuit legalization algorithms honor the horizontal cell order of GP. A trivial extension is to treat the maximum height of cell as the row height (i.e., to expand all cells to the same height as the highest cell), which obviously wastes much space. The difficulty otherwise is that simply shifting cells in a Abacus-like manner can form relatively big blanks in a cluster of multi-row-high cells that are not big enough to accommodate single-row-high cells. What is even worse is that shifting a multi-row-high cell in one row can induce overlap in another row. Referring to those blanks as *dead-space*, C.-H. Wang et al. carefully extended Abacus and optimized a quadratic programming (QP) of multi-row cells where numbers of cell pins were used as the weight of quadratic cell displacement to restrict wirelength overhead. They legalized cells from left to right and evaluated the legalization cost of each cell for every neighboring row in which displacement and dead-space area are considered (C.-H. Wang et al., 2017). Depending on the positions of four most-recently-legalized cells, different strategies are conducted for the legalization of the next cell among which six are discussed in detail so this method is more time-consuming compared with others.

Observe that when applying Karush-Kuhn-Tucker (KKT) condition on constrained

QPs of the form

$$\min_{x} \; c^\top x + \frac{1}{2} x^\top B x, \tag{2.1}$$

$$\text{s.t.} \; Ex \geq d, \tag{2.1a}$$

$$x \geq 0, \tag{2.1b}$$

where $B \in \mathbb{R}^{2m^2 \times 2m^2}$ is a symmetric positive-definite matrix and $E \in \mathbb{R}^{m^2 \times 2m^2}$ is a matrix of full row rank, the result is a a linear complementary problems (LCPs)

$$w = A \begin{pmatrix} x^\top & y^\top \end{pmatrix}^\top + \begin{pmatrix} c^\top & -d^\top \end{pmatrix}^\top, \tag{2.2a}$$

$$A = \begin{pmatrix} B & -E^\top \\ E & 0 \end{pmatrix}, \tag{2.2b}$$

$$\begin{pmatrix} x^\top & y^\top \end{pmatrix} w = 0, \tag{2.2c}$$

$$w \geq 0, \tag{2.2d}$$

$$x \geq 0, \tag{2.2e}$$

$$y \geq 0. \tag{2.2f}$$

These LCPs may be efficiently solved by modulus-based matrix splitting iteration method (MMSIM) which originally only guarantee convergence when the system matrix A of LCP is positive-definite (Bai, 2010). However, A here is obviously just positive-semi-definite. Besides, in the fixed-order legalization problem optimizing quadratic displacement, the relationship of cell order E is not full rank when multi-row cells are included. J. Chen et al. relaxed the constraint of right boundary and divided multi-row cells into multiple single-row cells to guarantee that E was full rank, and thus the feasibility of KKT condition was secured (J. Chen et al., 2017). The adjacency of split cells was guaranteed by Lagrangian relaxation and the cells placed out-

side the right boundary were finally resolved by a Tetris-like method (D. Hill, 2002), resulting in large maximum displacement for congested designs with irregular fence shapes.

Instead of fixing the row assignment by global placement positions, X. Li et al. assigned rows by a mixed integer quadratic program (MIQP) to resolve the effect of local dense area and relaxed the non-overlapping and site-alignment constraints with integer variables to handle large-scale circuits. When solving the relaxed LCP with MMSIM, the positive-definite requirement of the system matrix A was fulfilled by adding a small identity matrix to the objective matrix of the LCP (X. Li et al., 2019). Hence, the robustness of the iterative method is improved. Zhu, Chen, et al. proposed a movement-aware cell reassignment method. After initially solving the quadratic legalization problem by MMSIM, disruptive cells were identified and reassigned to sparse areas (Zhu, Chen, et al., 2020). MMSIM was then repeatedly conducted until the objective function had no further improvement. However, the solution space is greatly reduced when maintaining the cell order of GP. Thus it may result in poor results especially for high-utilization designs.

2.1.3 Window-based Placement

The third type of mixed-cell-height legalization methods are free from the artificial restriction on cell order. Chow et al. proposed a multi-row local legalization (MLL) algorithm where the cell list was traversed and the unplaced cells were legalized sequentially, as shown in Algorithm 2.2. In MLL, the concept of *segment* was introduced as a consecutive part of row that was not occupied by macros or placement blockages and belonged to the same fence region (Chow et al., 2016). When legalizing a target cell c_i, the segment assignments and the relative order of previously legalized cells are maintained. In a window surrounding the GP location of the target cell, different row assignments and insertion points of the target cell are explored and the total dis-

Algorithm 2.2 Legalization (Chow et al., 2016)

Require: Cell list $C = \{c_i\}_i$, global placement positions $\{(x'_i, y'_i)\}_i$.
Ensure: Legalized positions $\{(x_i, y_i)\}_i$.

```
Unplaced cell list U := C;
k := 0;
while U ≠ ∅ do
    for all c_i ∈ U do
        result := MLL(c_i, x'_i, y'_i, k);
        if result = Success then
            Move cells;
            U := U\{c_i};
        end if
    end for
    k := k + 1;
end while
return Legalized positions {(x_i, y_i)}_i;
```

placement is minimized by horizontally shifting cells that are completely in the local region. For each traversal iteration k, several tricks are triggered to enlarge the search spacing, such as larger local region and randomized target position. Its major limitation is that the later a cell is placed, the higher the probability that the cell will be placed with large displacement. In addition, the minimized displacement is w.r.t. the current locations of the local cells, which can be accumulated to large displacements w.r.t. the original global placement locations after many iterations.

Hung et al. divided the design into bins and reallocated cells from overfilled bins to under-filled bins by a network-flow-based method. With the feasibility guaranteed, an integer linear programming (ILP) is formulate for each bin to optimize the total displacement (Hung et al., 2017). After obtaining a legal solution, the bin boundaries are removed and the total displacement is further improved in a linear programming (LP) by shifting cells horizontally while keeping their relative order. Although parallelization across bins is implemented, the method is still time-consuming due to the high complexity of ILP. MrDP (Y. Lin et al., 2018) proposed a wirelength-driven

legalization based on a chain move scheme and extended a dual min-cost flow (MCF) method (Vygen, 1998) from single-row to multi-row cells for post-refinement. A potential problem is that using half-perimeter wire-length (HPWL) instead of displacement as objective function in legalization may disturb other objectives optimized in global placement.

2.1.4 Placement with Constraints

There are some recent works that legalize mixed-cell-height circuits with additional constraints. The wire resistance increases dramatically in sub-10 nm technologies, which cases longer delay and more significant supply voltage (IR) drop (Besser, 2017). Heo et al. conducted dynamic-programming-based double-row detailed placement with the robustness of power delivery network (PDN) improved by power staple insertion in post-placement flows (Heo et al., 2019). Zhu, Huang, et al. introduced a six-track cell library with two types of power and ground (P/G) alignment. Cell types with the original type of P/G alignment have P/G pins at the top and bottom of cells while double-row cell types with the new type of alignment named *PPNN* have P/G pins at one-fourth and three-fourth of the cell height and thus are aligned at half-rows (Zhu, Huang, et al., 2020). MMSIM is extended to handle the half-row fragmentation at the top and bottom of PPNN cells and the minimum width (MW) constraint of consecutive PPNN cells. However, none of the previous works solve the problem comprehensively. Some of them target at minimizing HPWL while some focus on the total displacement but none of them simultaneously handle other important measures and constraints like the existence of fence region and routability issues which include pin inaccessible, pin short and edge spacing.

2.2 Detailed Routing

Rule-based routing frameworks are friendly to conventional design flow and have been used for decades. These detailed routing frameworks can be classified into grid-based and grid-less depending on the utilized routing models. We acknowledge that grid-less frameworks are also investigated in literature (Y.-H. Lin et al., 2012) but their recent use is limited to pin access and grid-based frameworks have gained popularity on regular wire routing for their pruned solution space and natural adherence to uniform pitch manufacturing. Previous works usually focus on three parts: letting pins escape from FEOL to access BEOL, searching paths to connect multi-pin nets or the decomposed two-pin nets, and checking design rules for rip-up-and-reroute.

2.2.1 Pin Access

A pin can be accessed by either extending a wire to the pin or stacking a via in the pin. Qiu and Marek-Sadowska (2012) proved that given an unlimited number of routing layers, a placement was always routable if at least one free pin position existed or two pins to be connected were neighbors on a 2-D grid. Observing that the number of required routing layers increased rapidly when pin density became very high, an empirical threshold of pin density was suggested for a commercial router. To access pins regularly aligned on Metal 1 tracks, Ozdal (2009) assigned a Metal 2 wire segment for each pin in Metal 1 by solving a network flow problem where each x-coordinate or pin was a vertex and edges were between coordinate and pin vertices or between consecutive coordinate vertices. The direction of an edge is from the vertex on the left to the vertex on the right and the gain of an edge between a coordinate and a pin is calculated by the preference for the pin to span wire segment to the coordinate. The capacity of each pin vertex is one so the only two edges with flow from or to the pin vertex indicate the ideal left and right coordinate of the corresponding segment

on Metal 2.

As a bottleneck of detailed routing, off-grid pin access is required to let misaligned pins escape from dense clusters. Nieberg (2011) computed possible pin access path candidates and selected the shortest paths from pins to grid points not violating any design rules to perform correct pin access. The cells with different clock routing or orientation was divided into classes and possible candidates of pin access paths for each class were pre-computed to exploit redundancy. Maßberg and Nieberg (2013) expanded fixed metal shapes by the required spacing and obtain the feasible region for the center-lines of pin access wires. Subject to a constraint that each segment of the path has to satisfy a given length, the pin was connected by the shortest path within the feasible region making use of an extended Hanan grid. However, for cell types with dense pins and intra-cell routing, there may be no valid grid point for pins to escape without design rule violations (W.-H. Liu et al., 2019). In such cases, 3D grid-less pin access need to be considered.

Kahng et al. (2020a) proposed a multi-level, standard cell-based, and instance-based pin access analysis framework. They analyzed the intra-cell pin accessibility of all cells with distinct cell types, orientations, or offsets to all track patterns that existed in the design exchange format (DEF). Each pin of a unique instance is assigned at least three access points. An access point that is accessible upward, stores all vias that are violation-free according to the design rule check (DRC) engine by Kahng et al. (2020b) and marks the most preferred via among them. It also marks any of the other four directions, i.e., west, east, south, and north, that the router can access it. Among these access points, an access pattern selects one access point per pin, such that the preferred via types from these access points are compatible. Finally, a similar dynamic-programming-based method selects access patterns with cluster-awareness. However, in the enumeration of unique instances, influences by other routing embodiments such as blockages, I/O (i.e., input and output) pins, and P/G

rails are not considered.

2.2.2 Path Search

Although there is literature on the construction of hexagonal (Samanta et al., 2011) and octilinear (Lee et al., 2020) Steiner tree, we limit the discussion of path search on rectilinear routing in this thesis due to its ease of manufacturing. The simplest path searching method is to divide a multi-pin net into two-pin nets and apply regular patterns including L-shape, Z-shape, and monotonic routing to each two-pin net. RegularRoute applies regular routing patterns layer-by-layer to avoid jogs and unnecessary detours (Y. Zhang & Chu, 2013). From the bottom layer, the routing in each panel is formulated as a maximum weight independent set (MWIS) problem. By assigning tracks in one layer, pins are promoted to another end of the tracks to continue routing in the upper layer.

More sophisticated methods use variants of Dijkstra (1959) single-source shortest path algorithm with A*-like heuristics of lower-bound cost estimation between the current searching node and a preferred target node (Hart et al., 1972). Without honoring routing blockages and guides, lower-bound π_H is fast calculated the Manhattan distance between the searching node and the closest target node (Hetzel, 1998). Blockage-aware lower-bound π_P is a more accurate estimation and thus reduces the number of labeling steps needed by Dijkstra algorithm but the calculation of π_P is time-consuming (Peyer et al., 2009). The on-track path search of BonnRoute uses π_H for most of time and π_P when a large detour occurs in the routing guide to trade-off between the number of labeling steps and the running time of lower-bound computing (Gester et al., 2013).

Introducing the concept of tunnel as a set of global routing guides connecting a two-pin nets, Gonçalves et al. (2019) propagated reference points identified on each routing guide from target to source and integrated a tunnel-aware lower-bound esti-

mation

$$\pi_G(n) = \min\{lb(n, r) + TL(r)\} \tag{2.3}$$

to achieve a closer behavior to depth-first-search (DFS), where r is a reference point in the same routing guide as n, lb is the Manhattan distance modified by wrong-way routing cost, and TL is the propagated tunnel-aware cost.

Another set of methods consider searching paths with ILP or Boolean satisfiability (SAT) solvers. For example, TritonRoute partitions each layer into parallel panels and formulates the routing problem on each panel as an ILP (Kahng et al., 2020b). This set of methods may be time-consuming and thus are more suitable for solving small-scale problems, e.g., intra-cell routing (Park et al., 2020).

2.2.3 Design Rule Awareness

Xu et al. (2016) proposed a pin-access-driven rip-up-and-reroute scheme under self-aligned double patterning (SADP) constraints. G. Sun et al. (2016) prevented parallel run length (PRL) by moving portions of interconnect to adjacent track. There are some recent works due to the ISPD 2018 initial detailed routing contest featuring ten designs on 32 nm or 45 nm node with up to 0.3 million standard cells and 0.2 million nets, and complex design rules including width-dependent spacing, end-of-line (EOL) spacing, cut spacing, and minimum metal area to ensure printable GDSII masks for nanometer circuit designs (Mantik et al., 2018). F.-K. Sun et al. (2018) assigned distinct grid points to pins by applying Hopcroft and Karp (1973) algorithm to make sure no consecutive grid points were assigned to the same pin, which was a heuristic to reserve possible access points for other pins. A negotiation-based track assignment was then conducted considering via violations. Dr. CU is an optimal path searching algorithm handling the minimum-area constraint (G. Chen, Pui, Li, & Young, 2019). Ding et al. (2020) added assisting routing blockages around original blockage with

PRL rules to guide detailed routing honoring rules and resolving violations.

The recent ISPD 2019 initial detailed routing contest introduced ten more designs on 32 nm, 45 nm, or 65 nm node with up tp 0.9 million standard cells and 0.9 million nets, and additional design rules including length-dependent PRL spacing, EOL spacing with parallel edges, and corner-to-corner spacing (W.-H. Liu et al., 2019). Most traditional detailed routers relies on post-processing to fix design rule violations. As design rules get tremendously more complex and abundant, post-processing fails more frequently.

2.3 Split Manufacturing

As gate delay scales down proportionally with the semiconductor feature size, interconnect, unfortunately, degrades proportionally with the square of the downsizing due to the scaling up of wire resistance and have dominated timing (Bohr, 1995). Security-oblivious physical design tools typically place interconnected components close to each other in the FEOL layers and wires them up through the BEOL layers using short paths (H. Li et al., 2018). While delivering effective designs in terms of power, performance, and area, such an approach leads to some information leakage so that, with an understanding of physical design tools, the structural information gathered from the FEOL layers can be utilized in the scenario of split manufacturing to infer the missing BEOL connections.

2.3.1 Split Manufacturing Attack

Rajendran et al. (2013) demonstrated the first naïve proximity attack, where they leveraged only the fact that interconnected modules are typically placed close by. They aimed to recover the BEOL by connecting every target pin to its closest candidate pin. The attack performed reasonably well for hierarchical designs with a few

nets between the modules but showed limited success for flat designs and large layouts. Instead of selecting a single candidate pin, Y. Wang et al. (2018) proposed an enhanced proximity attack based on a network-flow model. While constructing a flow graph, they set a weighted sum of the proximity on preferred and non-preferred routing directions as the edge cost, and adopted driving capacitance as the edge capacity. However, the network-flow formulation was relaxed to the naïve proximity attack once cell libraries had loose capacitance constraints. The attack performs iterative edge removal when loops occur, which causes significant running time.

In another framework, Magaña et al. (2017) identified a relatively small list of candidates within a search neighborhood based on physical proximity, utilizing place and routing information. Zeng et al. (2019) analyzed the security of split manufacturing on industrial designs with random-forest classifier. However, their classifiers do not predict the BEOL connections directly, but generate a list of candidates with considerable size instead. For instance, when attacking layouts split after the fourth metal layer (M4), i.e., the FEOL contains M1 – M4, their most successful classifier provides on average several hundreds or even thousands of candidates for each broken connection. Here it can become practically impossible to retrieve all correct connections.

Besides traditional classifiers, we believe that deep learning is a good match for attacking split manufacturing because attackers may have access to an extensive database of designs to train on. Among other applications, deep learning has improved its accuracy and capacity in game playing (Silver et al., 2018), object detection and segmentation (Griffiths & Boehm, 2019), routability prediction (Zhou et al., 2019), and design-for-manufacturability (Ma et al., 2020). We caution that attacking split manufacturing with deep learning entails handling a large variety of data. Vector-based data are ranging, e.g., from signed gate displacements to unsigned wirelengths and from integral pin counts to floating point load capacitance. Additionally, layout images can represent routing segments and their directions as well as congestion.

Such image-based data naturally constitute rich information that can be useful for an advanced attack. One limitation of current neural network architectures is that they can only handle either vector- or image-based features, but not both types together. Accordingly, one challenge for our work is to combine both vector-based and image-based features in a unified framework. Another limitation when applying deep learning is that traditional two-class classifiers can only predict the probability of each possible BEOL connection, but not representing the physical-design reality that each sink pin is assigned to exactly one source. When selecting the source with the largest predicted connection probability, traditional classifiers can be easily misled by outlying predictions of negative samples.

2.3.2 Split Manufacturing Defense

According to the various attack methods, several defense algorithms were proposed to escalate the security of split designs. Most of the defenses artificially promote more nets to upper layers during routing to create more candidates in the neighborhood of broken connections to complicate the proximity attack. Magaña et al. (2017) added artificial routing blockages to the designated split layer after global routing, and performed global routing again, to make the routing tools elevate more wires over the split layer. The artificial blockages were removed before executing detailed routing, to guarantee routability. However, we note that layer assignment in global routing is not necessarily in accordance with detailed routing (W.-H. Liu et al., 2019) as the routing guides passing to detailed router can occupy multiple layers and the router may route out of guides when the designs are too congested. Therefore, wires elevated above split layer in global routing are possibly routed in the split layer or even below, which undermines the controllability and effectiveness of the protection scheme. Moreover, the scheme proposed by Magaña et al. (2017) can control the length/size of blockages only at the lower left corner of each routing-grid bin, which limits the solution space

for routing perturbation. A more sophisticated defense compatible with industry-standard tools and physical design flow was proposed by Y. Wang et al. (2017) including techniques of layer elevation, detour, and decoy, where wires to be detoured and decoyed are selected based on IC testing principles. Patnaik, Knechtel, et al. (2018) lifted wires to the BEOL to reduce the percentage of netlist recovery (PNR) without sacrificing routing resources in lower layers. However, severe wirelength overheads were incurred to achieve the proposed security criterion.

Another set of defense attempts extend the perturbation during placement or even include dummy cells. Sengupta et al. (2017) colored cells by connection or by type and placed cells with the same color into the same fence. These two placement techniques can significantly reduce the correct connection rate with the cost of more than doubled wirelength. Y. Wang et al. (2018) developed a security-driven placement-perturbation algorithm by obfuscating the cell placement based on the layer-assignment result after global routing. However, their placement perturbation caused large wirelength overheads. Assuming that attackers hold the gate-level netlist of the designs, M. Li et al. (2018) proposed a secure-by-construction split manufacturing flow by converting wire lifting and cell insertion into a mixed-integer linear programming (MILP) and solving it with a Lagrangian relaxation algorithm and a minimum-cost flow transformation. However, severe area overheads were incurred to achieve the proposed security criterion.

□ End of chapter.

Chapter 3

Multi-row Cell Legalization

Summary

In this chapter, we propose a legalization method for mixed-cell-height circuits by a window-based cell insertion technique. Compared with the champion of the 2017 Contest at the International Conference on Computer Aided Design, our algorithm achieves 35% and 13% less average and maximum displacement respectively as well as significantly fewer routability violations. Comparing our algorithm with the state-of-the-art algorithms on this problem, there is an 8% improvement in average displacement with comparable maximum displacement.

Placement is one of the most critical stages in the physical synthesis flow. Traditionally, placement consists of three steps: global placement, legalization, and detailed placement. With macros fixed, only standard cells are considered in the legalization step. Besides providing an overlap-free solution close to the global placement solution, design rules on power and ground (P/G) alignments, fence region, and routabil-

ity (e.g., edge spacing, pin short/inaccessible) should be considered. In sub-seven-nanometer technology nodes, the gap between the device and metal pitches becomes larger, the number of routing tracks becomes smaller and thus more standard cells are implemented with multi-row height (Park et al., 2020). It is invalid to treat all the multi-row cells as macros or standard cells. If they are treated as fixed macros, the solution space is lost and degradation occurs on the objective. If they are regarded as standard cells, shifting a cell in one row may cause cell overlaps in another row and thus traditional legalization algorithms are infeasible. In this chapter, we present a fast and high-quality legalization framework for standard cells with mixed cell heights, which outperforms the state-of-the-art under the displacement objective. Our legalizer optimizes both the maximum and average displacement. It also considers fence constraints and routability constraints (e.g., edge spacing and pin access/short), minimizing the number of violations. Our major contributions are as follows.

- We develop a mixed-cell-height circuit legalizer optimizing the maximum and average displacement with constraints on fences, edge spacing, and pin accessibility.
- We devise a thread-safe method called multi-row global legalization (MGL) that inserts a cell optimally into a window, minimizing the average displacement of all the cells in the region from their global placement instead of their current positions.

The rest of this chapter is organized as follows. Section 3.1 illustrates the problem formulation and constraints. Section 3.2 provides a detailed explanation of our proposed techniques, followed by an effectiveness verification of our approach in Sec. 3.3.

3.1 Problem Formulation

Given a set of m multi-row height cells $C = \{c_1, c_2, \cdots, c_m\}$. The cell height and displacement are measured in terms of the multiple of single row height. Let $\mathcal{H}$ be the set of cell height, $C_h \subseteq C$ be then set of cells whose height is h. The problem is to place each cell c_i from global placement (x'_i, y'_i) into (x_i, y_i) with a corresponding displacement:

$$\delta_i = \delta_{x_i} + \delta_{y_i} = |x_i - x'_i| + |y_i - y'_i|, \tag{3.1}$$

such that the maximum and average displacement is minimized. Here, the average displacement S_{am} is weighted by the number of cells of the same height:

$$S_{am} = \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} \frac{1}{|C_h|} \sum_{c_i \in C_h} \delta_i, \tag{3.2}$$

which is the metric used in the 2017 Contest at the International Conference on Computer Aided Design (ICCAD) (Darav et al., 2017).

Besides, cells should be overlap-free and aligned to placement sites of the chip. The power and ground (P/G) alignment and the fence region constraint are treated as hard constraints:

- Cells with even cell heights must be placed in alternate rows with aligned P/G rails (Chow et al., 2016);
- Cells assigned to a fence region must be exclusively placed inside the fence boundary (Bustany et al., 2015).

Note that there is no restriction on the row assignments for odd-row-high cells just like single-row-high cells as long as they are flexible to be flipped vertically to correctly align with the P/G rails.

We divide the routability violations of cell pins in two categories and consider them as soft constraints. Inter-cell pin short or spacing violations are modeled as

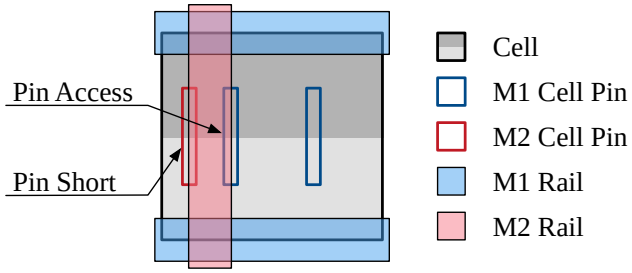


Figure 3.1: Pin access and pin short.

edge spacing. Violations with input and output (I/O) pins and P/G grids are modeled as pin short or pin access violations:

- While each cell edges belongs to an edge type, a minimum spacing is required between any two cell edge types (Yutsis et al., 2014);
- Signal pins of cells should be able to connect with vias or regular wires without creating violations. If a signal pin is short or inaccessible due to the P/G grids and input/output (I/O) pins, there may not be available access point for the pin in detailed routing (H. Li, Chen, et al., 2019).

In modern chip design, the P/G rails are usually regular grids running horizontally or vertically in alternate metal layers. Note that a signal pin on metal layer k is short if it overlaps with a P/G rail or an I/O pin on metal layer k. A signal pin is inaccessible if it cannot be accessed by a regular wire without creating violations on metal layer k or it cannot be accessed by a via without creating violations on metal layer $k + 1$. As shown in Fig. 3.1, the left pin on metal layer one (M1) has pin access problem with the rail on metal layer two (M2), and the M2 pin is short with the M2 rail.

3.2 Algorithms

Given a global placement solution with multi-cell-height circuits, we legalize it by multi-row global legalization (MGL) which inserts the cells sequentially into the placement region. Note that a cell may belong to a specific fence region. Cells that do not belong to any fence regions should be placed in the *default fence* region which is the region outside all other given fence regions.

3.2.1 Legalization

In this section, we will introduce the MGL method which legalizes cells sequentially to minimize the average and maximum displacement from the given global placement positions.

Inspired by MLL (Chow et al., 2016), MGL first regards all unfixed cells as unplaced cells and takes these cells out from the core area. By the descending order of cell area, MGL then sequentially legalizes cells in local windows centering their global placement positions. Different from MLL that calculates displacement based on the current cell locations and can eventually accumulate a large displacement w.r.t. global placement locations, MGL minimizes the displacement from global placement locations directly. Figure 3.2 illustrates an example, where the given global placement positions are shown in Fig. 3.2(a). Suppose cells $c_1 - c_4$ are legalized before target cell c_t is inserted as in Fig. 3.2(b), which already has a total displacement of two. In Fig. 3.2(c), MLL optimizes the total displacement w.r.t. current locations and achieves a value of one. However, the total displacement from global placement position is actually three. Figure 3.2(d) shows the result with minimized total displacement from global placement positions (i.e., two) produced by MGL.

Algorithm 3.1 shows the flow of MGL. When legalizing a *target cell* c_t, a *window* r_t around its global placement position (x'_t, y'_t) is considered. Meanwhile, legalized cells

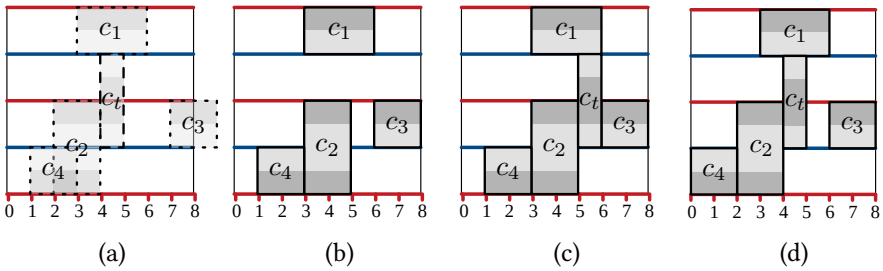


Figure 3.2: Comparison between multi-row local legalization (MLL) and our purposed multi-row global legalization (MGL): (a) Global placement; (b) Four cells legalized; Final results of minimizing the total displacement w.r.t. (c) current locations (MLL) and (d) global placement locations (MGL).

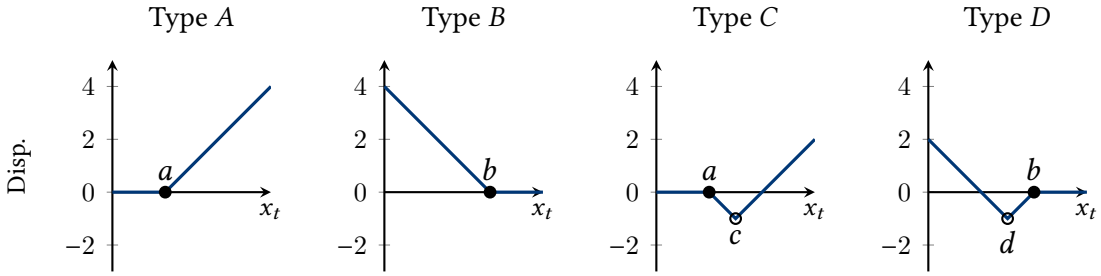


Figure 3.3: Four types of displacement curves.

Algorithm 3.1 Multi-row Global Legalization (MGL)

Require: Window r_t, global placement position (x'_t, y'_t) of target cell c_t.
Ensure: Legal positions of c_t and local cells.

```
1: Find candidate insertion points 𝒫 in r_t;
2: for all p_i ∈ 𝒫 do
3:     for all breakpoint b do
4:         Store (x_b, left slope of b, right slope of b) in points;
5:     end for
6:     AcCurve(points);                          ▷ Accumulate slopes.
7:     d_i ← optimal displacement;
8:     x_t^i ← optimal x-coordinate;
9:     y_t^i ← y-coordinate of p_i;
10: end for
11: j ← arg min_i d_i;
12: Place c_t at (x_t^j, y_t^j) and spread local cells;
```

that lie completely within r_t are referred as *local cells*, which can be shifted for legalizing c_t. In MGL, the row and order assignment of local cells are fixed, but those of c_t are enumerated and evaluated. With row and relative order of local cells fixed, inserting c_t with height h_t implies that we need to place c_t in some gaps between the legalized cells in h_t consecutive rows. A combination of those gaps for inserting c_t is an *insertion point*. For a given c_t and r_t, using the enumerating method by Chow et al. (2016), MGL first obtains all the legal insertion points (line 1). It then calculates the optimal displacement cost of all the insertion points (lines 2 to 10). The displacement curve, which represents the cost of each insertion point with varied x-coordinate of c_t, can be constructed by adding up the all displacement curves of the local cells and c_t. The construction of the curves is explained with more details later. The best position to insert c_t with an insertion point is the position with the lowest cost on the displacement curve. After inserting c_t at the position with the lowest cost, local cells are shifted to the left or right when needed to legalize the placement (lines 11 to 12). If there is no valid insertion point, the size of r_t is increased and the legalization for c_t is conducted again. In practice, the window size is initialized as the size of c_t, i.e., to check if c_t can be directly placed at its global placement position without creating violations. After that, r_t expands horizontally by six rows and vertically by four rows for each iteration.

In MLL, when a valid insertion point p is considered, there are only two types of displacement curves for any local cell as illustrated by type A and B in Fig. 3.3 in which the horizontal axis is the x-position of the target cell, the vertical axis is the displacement contributed by the local cell, and the solid dot indicates the x-position of the target cell starting to push the local cell. The curves are of these shapes since we are measuring the distance from the original positions of the local cells before the target cell is inserted. Cells on the right of p in the window have displacement curves of type A because they may be pushed to the right of their original positions

due to the insertion of target cell at different horizontal positions. Similarly, cells on the left of p in the window have displacement curves of type B because they may be pushed to the left due to the insertion. The solid dots of these curves are called *critical positions* as they start contributing to the cost in displacement if the target cell moves beyond the corresponding critical coordinate (Chow et al., 2016). Since there are only these two types of curves in MLL, the optimal position to place the target cell can be obtained efficiently by finding the median of all these critical positions.

In MGL, the scenario is more complicated since the displacement is calculated w.r.t. the given global placement position instead of the current position. There are two more types of displacement curves as illustrated by types $A - D$ in Fig. 3.3 in which the circle marks indicates the x-position of the target cell pushing the local cell over its global placement position. Without loss of generality, we first discuss the local cells on the right of a valid insertion point p, where there are two possible types of curves A and C. Local cells with their global placement positions at or on the left of their current positions have displacement curves type A because the target cell will only push the local cells further to the right from their global placement positions. For local cells with their global placement positions on the right of their current positions will have displacement curve type C. The turning points on these curves are either critical positions as in MLL (labeled by a) or circle marks (labeled by c). We call all these turning points *breakpoints*. Similarly, for local cells which are on the left of the valid insertion points, their displacement curves will be of types B and D. Considering cells in Fig. 3.2(b), the displacement curve of c_2 is of type D while being pushed left and the displacement curve of c_4 is of type C while being pushed right.

To prove the property of displacement curves, we introduce the definition of clusters as follows:

Definition 3.1. A **right cluster** is a set of cells that can move to the right together

without creating overlaps with other cells. Specifically, a cell without adjacent cell on the right is a right cluster by itself.

Property 3.1. Every right cluster has a cell without adjacent cell on its right so that this cell is a right cluster itself.

Property 3.2. Every right cluster has a cell without adjacent cell in the cluster on its left so that the cluster without this cell is still a right cluster.

Lemma 3.1. If all cells in a window are placed at their optimal positions, the displacement curve for moving a right cluster to the right is piece-wise linear, non-decreasing, and convex.

Proof. Base on the two properties, we prove it by Mathematical Induction. In the base step, if a single cell is a right cluster, the global placement position of this cell will not be on the right of its current position because otherwise, we will have moved it further to the right. Therefore, its displacement curve is like type A which is piece-wise linear, non-decreasing, and convex. In the induction step, we assume that the lemma is correct for all right clusters of $r-1$ cells. Then for any right cluster $\mathcal{R}$ of r cells, we split it into a single cell c_l without adjacent cell in $\mathcal{R}$ on its left and another right cluster $\mathcal{R}\backslash\{c_l\}$ with $r-1$ cells. If the displacement curve d_l of cell c_l is of type A, the lemma is clearly correct for $\mathcal{R}$ because adding two piece-wise linear, convex, and non-decreasing curves will result in the same properties. If d_l is of type C, then its slops is

$$d_l' = \begin{cases} 0, & x_t < a, \\ -n_l, & a < x_t < c, \\ n_l, & c < x_t, \end{cases} \tag{3.3}$$

where n_l is the weight on the displacement of cell c_l. Since all the cells in $\mathcal{R}$ are originally placed at their optimal positions, the slope of the displacement curve $d_{\mathcal{R}}$ of

the cluster $\mathcal{R}\backslash\{c_l\}$ satisfies

$$\begin{cases} d'_{\mathcal{R}} = 0, & x_t < a, \\ d'_{\mathcal{R}} \geq n_l, & a < x_t, \end{cases} \tag{3.4}$$

because otherwise, the whole right cluster $\mathcal{R}$ should have been moved further to the right. Hence, adding up d_l and $d_{\mathcal{R}}$ gives a piece-wise linear, non-decreasing, and convex shape curve. Therefore, the lemma is correct for any right cluster of r cells, which accomplishes the proof. □

Theorem 3.1 states that the final displacement curve is convex if the local cells are originally at their optimal positions w.r.t. their global placement positions, before the target cell is inserted.

Theorem 3.1. Consider a window W containing a target position (x'_t, y'_t), a set $\mathcal{S}$ of local cells lying completely inside W and a target cell c_t to be inserted into W. If all the cells in $\mathcal{S}$ are originally placed at their optimal positions (total displacement is the smallest under the fixed row and fixed order constraint) w.r.t. their global placement positions, the displacement curve, where the x-axis is the position of the target cell x_t, obtained by adding up the displacement curves of all the cells in $\mathcal{S}$ is piece-wise linear and convex.

Proof. The total displacement curve is clearly piece-wise linear because the displacement curve for each cell is piece-wise linear and $|\mathcal{S}|$ is finite. The right part of the displacement curve is accumulated by a set of right clusters so it is non-decreasing and convex. Similarly, the left part of the displacement curve is non-increasing and convex. Therefore, the total displacement curve is piece-wise linear and convex. □

The pre-condition of having the local cells at optimal positions w.r.t. their global placement positions would require running a min-cost flow (MCF) before invoking MGL that will lengthen the running time. Therefore, in our implementation, we will

Algorithm 3.2 Accumulate Slopes (AcCurve)

Require: $points = \{(x_b, k_b^l, k_b^r) \mid b \text{ is a breakpoint}\}$.
Ensure: Total displacement curve.
1: Sort $points$ by x_b;
2: **for all** $x_{b_1} = x_{b_2}$ **do**
3: $k_{b_1}^l \leftarrow k_{b_1}^l + k_{b_2}^l$;
4: $k_{b_1}^r \leftarrow k_{b_1}^r + k_{b_2}^r$;
5: Remove b_2 from $points$.
6: **end for**
7: **for all** breakpoint b **do**
8: $\bar{k}_b^l \leftarrow \sum_{x_i < x_b} k_i^r + \sum_{x_i \geq x_b} k_i^l$;
9: $\bar{k}_b^r \leftarrow \sum_{x_i \leq x_b} k_i^r + \sum_{x_i > x_b} k_i^l$;
10: **end for**

compute the cost at each breakpoint to find the optimal position. Since the number of breakpoints is linear with the number of local cells, the optimal positions can be found in linear time as shown in Algorithm 3.2. Given a set of breakpoints sorted by x-coordinate (line 1), we first merge breakpoints sharing the same x-coordinate by adding the slopes together (lines 2 to 6). For the total displacement curve, its left (right) slope at each breakpoint is the right slope sum of left (non-right) breakpoints adding the left slope sum of non-left (right) breakpoints. Then the optimal x-coordinate x_{b_0} is at the boundry of its feasible region or satisfies $\bar{k}_{b_0}^l \bar{k}_{b_0}^r \leq 0$.

3.2.2 Multi-thread Implementation

In this section, we will introduce the multi-thread implementation of MGL. Recall from the previous section that MGL of a target cell is conducted within a local window centering the global placement position of the target cell. Since only legalized cells that are completely inside the local window are movable, local windows that do not overlap with each other can be processed simultaneously. A scheduling step decides which local windows can be processed at the same time. The *scheduler* maintains a list $\mathcal{L}_p$ containing 2-tuples of target cell and its corresponding local window under

processing. In each iteration, the scheduler selects a fixed number of these 2-tuples that do not have their local windows overlapping with each other and pushes them into $\mathcal{L}_p$.

Legalizers in the child threads process MGL whenever there is unprocessed 2-tuple in $\mathcal{L}_p$ and apply the legalization step. The scheduler switches to perform MGL as well when it finished scheduling. If MGL failed to insert the target cell, the local window is expanded and the new 2-tuple of target cell and expanded local window will then be pushed into a waiting list $\mathcal{L}_w$ from which the scheduler will select the 2-tuples for the next iteration. Since the scheduler synchronizes all threads after each iteration and the cell order in $\mathcal{L}_w$ is kept the same as the original cell order, the multi-thread implementation is deterministic once the capacity of list $\mathcal{L}_p$ is determined.

3.2.3 Routability-driven Refinement

Edge spacing rules define the minimum distances between different types of cells. For the MGL, there are two kinds of cell moves. One is to horizontally shift cells to the left or right with row assignment and cell order fixed, where the edge spacing is reserved between each pair of consecutive cells. The other is to insert a target cell, where the edge spacing is reserved when constructing insertion points. For example, the left-most possible position for cell c_r is calculated by

$$x_r = x_l + w_l + e_{l,r}, \tag{3.5}$$

where x_l and w_l are the x-coordinate and width of the cell c_l on its left and $e_{l,r}$ is the edge spacing requirement for cell pair (c_l, c_r).

Pin access and pin short violations are caused by signal pins of cells sheltered by P/G rails or I/O pins so that no same layer wire or upper layer via can access the pins without causing design rule violations. The source of violations can be divided in

three types: overlaps with horizontal rails, with vertical rails and with I/O pins. Due to limited numbers of lower layer I/O pins and resource reservation for their routing, the regions below I/O pins in metal layer 1 (M1) or metal layer 2 (M2) are treated as placement blockages.

In MGL, these three types of violations are considered separately. When enumerating the insertion points, local cells are moved to the left-most and right-most positions in the local region to validate the insertion points. After a local cell being moved to its left-most position, it is moved back to the right until it has no violation with a vertical rail; the same happens for right-most positions. If an insertion point has violation with a horizontal rail, it will not be considered as a valid insertion point. While evaluating an insertion point, the optimal position is chosen accordingly to the displacement curve. If there is violation with a vertical rail, positions on the left or on the right will be considered until a least-displaced-position without any violation is found. For I/O pins, penalties will be given to the insertion points that overlap with I/O pins.

3.3 Experimental Results

We implemented the proposed legalization algorithm for mixed-cell-height circuits in C++ programming language. We run all experiments on a 64-bit Linux machine with eight cores of Intel Xeon 2.1GHz CPUs and 64GB RAM.

3.3.1 ICCAD 2017 Contest Benchmark Results

In the first experiment, we compare with a binary from the first place of ICCAD 2017 Contest in Multi-Deck Standard Cell Legalization (Darav et al., 2017) and a binary from the state-of-the-art work (Zhu, Chen, et al., 2020). We do not explicitly compare with (X. Li et al., 2019) because the average displacement in its objective func-

Table 3.1: Statistics of ICCAD 2017 Contest Benchmarks

Benchmark	#Cells of Different Heights				Density	HPWL
	1	2	3	4		(+e9)
des_perf_1	112644	0	0	0	90.6%	1.22
des_perf_a_md1	103589	4699	0	0	55.1%	2.16
des_perf_a_md2	105030	1086	1086	1086	55.9%	2.18
des_perf_b_md1	106782	5862	0	0	55.0%	2.11
des_perf_b_md2	101908	6781	2260	1695	64.7%	2.14
edit_dist_1_md1	118005	7994	2664	1998	67.4%	4.01
edit_dist_a_md2	115066	7799	2599	1949	59.4%	5.10
edit_dist_a_md3	119616	2599	2599	2599	57.2%	5.33
fft_2_md2	28930	2117	705	529	82.7%	0.45
fft_a_md2	27431	2018	672	504	32.3%	1.09
fft_a_md3	28609	672	672	672	31.2%	0.95
pci_bridge32_a_md1	26680	1792	597	448	49.5%	0.45
pci_bridge32_a_md2	25239	2090	1194	994	57.7%	0.57
pci_bridge32_b_md1	26134	1756	585	439	26.6%	0.66
pci_bridge32_b_md2	28038	292	292	292	18.3%	0.58
pci_bridge32_b_md3	27452	292	585	585	22.2%	0.58

tion does not consider the weights on cell numbers as Eq. (3.2) and thus the results is biased toward single-row cells since, shown in Table 3.1, at least 85% cells are single-row cells. Some other benchmark statistics are also shown in Table 3.1 including design density and the HPWL of global placement. Here, *density* is measured by the total cell area over the total free area.

We evaluate the pin access violations with Cadence Innovus 18.12 taking both *Pre-route Design Rule Check (DRC) Violations* and *Pin Access Violations* into count. We adopt the score function in the contest as much as possible to have a more comprehensive comparison:

$$S = \left(1 + S_{hpwl} + \frac{N_p + N_e}{m}\right)\left(1 + \frac{\max_i\{\delta_i\}}{\Delta}\right) S_{am}, \tag{3.6}$$

where S_{hpwl} is the ratio of HPWL increase, N_p and N_e are the numbers of violations

Table 3.2: Comparison on ICCAD 2017 Contest Benchmark

Benchmark	Avg. Disp.			Max. Disp.			HPWL (+e9)			Pin Inaccessible			Edge Space			Score S			Runtime (s)		
	1st	TCAD†	Ours	1st	TCAD†	Ours	1st	TCAD†	Ours	1st	TCAD†	Ours	1st	TCAD†	Ours	1st	TCAD†	Ours	1st	TCAD†	Ours
des_perf_1	0.710	0.807	0.870	7.7	6.6	10.1	1.30	1.33	1.34	7313	3636	216	0	0	0	0.87	0.96	1.05	9.55	22.64	28.91
des_perf_a_md1	1.818	0.994	0.913	62.6	60.7	60.7	2.26	2.23	2.23	2566	3080	2	0	0	0	3.16	1.69	1.51	5.46	3.80	3.85
des_perf_a_md2	3.476	1.328	1.143	68.0	40.4	48.1	2.28	2.26	2.25	2604	2986	2	0	0	0	6.26	1.98	1.75	5.50	6.27	3.84
des_perf_b_md1	0.698	0.701	0.659	9.0	9.0	11.4	2.16	2.16	2.16	2442	2392	17	0	0	0	0.80	0.80	0.75	4.47	3.28	4.34
des_perf_b_md2	0.655	0.655	0.617	20.0	19.4	24.1	2.19	2.20	2.19	2125	2120	0	0	0	0	0.82	0.82	0.78	4.16	2.40	3.80
edit_dist_1_md1	0.798	0.765	0.660	7.9	7.8	5.8	4.09	4.10	4.09	3435	3414	0	0	0	0	0.90	0.87	0.71	5.17	3.10	3.89
edit_dist_a_md2	0.646	0.639	0.612	16.4	16.4	16.4	5.18	5.18	5.17	3230	3235	0	0	0	0	0.78	0.77	0.72	4.31	3.71	4.29
edit_dist_a_md3	0.901	0.838	0.760	28.0	23.3	23.3	5.46	5.48	5.46	3479	3361	194	0	0	0	1.21	1.09	0.96	44.34	25.66	16.73
fft_2_md2	0.675	0.837	0.716	6.6	7.2	6.3	0.49	0.52	0.51	1942	932	75	5980	0	0	0.96	1.07	0.86	1.18	0.90	1.40
fft_a_md2	0.566	0.568	0.562	34.3	34.3	34.3	1.11	1.11	1.11	807	814	11	0	0	0	0.79	0.79	0.77	0.99	0.43	0.87
fft_a_md3	0.536	0.538	0.529	11.0	11.0	11.0	0.97	0.97	0.97	764	765	5	0	0	0	0.62	0.62	0.60	1.02	0.46	0.80
pci_bridge32_a_md1	0.696	0.664	0.650	42.6	42.6	45.7	0.47	0.47	0.48	588	564	1	0	0	0	1.05	1.00	0.99	1.11	0.89	1.07
pci_bridge32_a_md2	0.898	0.894	0.832	27.2	18.1	18.1	0.59	0.60	0.60	1845	813	18	0	0	0	1.27	1.15	1.05	1.89	3.02	1.30
pci_bridge32_b_md1	1.064	0.824	0.776	87.7	51.4	51.4	0.69	0.68	0.68	587	753	0	0	0	0	2.11	1.32	1.21	1.26	0.63	1.12
pci_bridge32_b_md2	1.084	0.791	0.698	72.3	54.6	61.7	0.60	0.59	0.59	646	862	0	0	0	0	1.97	1.30	1.17	1.08	0.65	1.23
pci_bridge32_b_md3	1.910	1.024	0.916	68.2	49.8	49.8	0.62	0.61	0.61	656	849	1	0	0	0	3.46	1.65	1.43	1.34	1.09	1.04
Norm. Avg.	1.35	1.08	**1.00**	1.13	**0.96**	1.00	1.02	1.02	**1.00**	398.09	445.95	**1.00**	—			1.47	1.09	**1.00**	1.21	**0.94**	1.00

† Zhu, Chen, et al. (2020).

on pin access/short and edge spacing, m is the number of cells, displacement δ_i and S_{am} are calculated by Eqs. (3.1) and (3.2) and Δ is 100 (Darav et al., 2017). Note that the runtime scores are not included because they are measured w.r.t other teams. The penalties of maximum displacement and target utilization are not included because their exact definitions are not clear and a constant factor is not revealed. The results are listed in Table 3.2 where our proposed algorithm achieves 35% smaller average displacement and 13% shorter maximum displacement compared with the first place. Our proposed algorithm also achieves 8% smaller average displacement and 4% longer maximum displacement compared with the state-of-the-art work. For routability-driven constraints, we have no edge spacing violations while the first place produces nearly six thousand in one case. We also have significantly fewer pin access violations. Without counting the designs that we are violation free, the numbers of violations are reduced by 398 times and 446 times on average, compared with the first place and the state-of-the-art, respectively. In terms of S, our purposed method has 47% and 9% improvement on average.

3.3.2 Modified ISPD 2015 Contest Benchmark Results

In the second experiment, we compare our method with three state-of-the-art placers (J. Chen et al., 2017; Chow et al., 2016; C.-H. Wang et al., 2017). The benchmarks are modified from the ISPD 2015 Blockage-aware Detailed Routing-driven Placement Contest (Bustany et al., 2015) and provided by Chow et al. (2016). Ten percent of the cells were selected and converted to double height and half width. To be consistent with other works listed in Table 3.3, we adapted our program to use total displacement as the objective function and ignored fences as well as routability-driven constraints. Note that the results of Chow et al. (2016) and C.-H. Wang et al. (2017) are improved ones reported by J. Chen et al. (2017). We can see that we have improvement over Chow et al. (2016) by 19%, over C.-H. Wang et al. (2017) by 16%, and over

Table 3.3: Comparison on Modified ISPD 2015 Contest Benchmark

Benchmark	#Cell	Density	Total Disp. (sites)				Runtime (s)			
			DAC16*	ASPDAC†	DAC17‡	Ours	DAC16*	ASPDAC†	DAC17‡	Ours
des_perf_1	112644	90.58%	279545	474789	242622	188719	6.1	7.5	2.4	2.7
des_perf_a	108292	42.90%	81452	73057	72561	71049	2.5	3.8	2.3	2.1
des_perf_b	112644	49.71%	81540	72429	71888	70959	2.2	3.9	2.3	2.3
edit_dist_a	127419	45.54%	59814	60971	62961	57264	1.8	4.9	2.8	2.3
fft_1	32281	83.55%	54501	53389	46121	38938	1.0	1.3	0.7	1.3
fft_2	32281	49.97%	25697	21018	20979	20381	0.4	1.1	0.6	0.6
fft_a	30631	25.09%	19613	18150	18304	17897	0.2	1.2	0.6	0.6
fft_b	30631	28.19%	28461	21234	21671	20852	0.4	1.2	0.6	0.6
matrix_mult_1	155325	80.24%	80235	73682	71793	61992	4.0	5.4	3.6	3.2
matrix_mult_2	155325	79.03%	75810	65959	65876	58250	4.2	5.4	3.7	3.1
matrix_mult_a	149655	41.95%	46001	40736	40298	39683	1.6	5.7	3.4	3.0
matrix_mult_b	146442	30.90%	40059	37243	37215	36658	1.2	5.6	3.2	3.2
matrix_mult_c	146442	30.83%	42490	40942	40710	39767	1.4	5.6	3.2	2.7
pci_bridge32_a	29521	38.39%	27832	26674	26289	25960	0.3	1.2	0.6	0.4
pci_bridge32_b	28920	14.30%	27864	26160	26028	26120	0.2	1.0	0.4	0.6
superblue11_a	927074	42.92%	1786342	1983090	1742941	1595907	29.7	50.3	26.3	22.2
superblue12	1287037	44.72%	2015678	1995140	1963403	1713915	103.6	56.5	38.6	31.6
superblue14	612583	55.78%	1599810	1497490	1566966	1330885	16.7	48.1	17.7	13.8
superblue16_a	680869	47.85%	1173106	1147530	1135186	1055668	20.7	41.8	18.7	14.9
superblue19	506383	52.33%	806529	808164	781928	705509	10.5	29.6	13.2	11.9
Norm. Avg.			1.19	1.16	1.08	**1.00**	1.04	2.09	1.08	**1.00**

* Chow et al. (2016) with improvement.
† C.-H. Wang et al. (2017) with improvement.
‡ J. Chen et al. (2017).

Table 3.4: Trade-off on Window Size

Benchmark	Avg. Disp.			Max. Disp.			Running Time		
	S*	M†	L‡	S*	M†	L‡	S*	M†	L‡
des_perf_1	0.962	0.964	0.968	48.4	49.5	69.8	3.166	4.742	8.342
des_perf_a_md1	0.937	0.919	0.940	92.7	60.7	60.7	0.903	1.809	3.735
des_perf_a_md2	1.237	1.148	1.142	49.7	48.1	48.1	0.772	1.667	3.230
des_perf_b_md1	0.677	0.675	0.676	52.4	53.8	47.1	0.821	1.280	2.144
des_perf_b_md2	0.626	0.618	0.617	37.1	27.3	26.5	0.709	1.307	2.618
edit_dist_1_md1	0.674	0.664	0.656	13.9	5.8	7.3	0.578	0.982	1.927
edit_dist_a_md2	0.620	0.614	0.613	24.1	16.5	16.4	0.711	1.301	2.772
edit_dist_a_md3	0.797	0.783	0.771	56.1	74.2	83.7	1.942	2.784	5.470
fft_2_md2	0.738	0.721	0.722	21.4	10.0	13.0	0.190	0.294	1.472
fft_a_md2	0.562	0.563	0.566	34.3	34.3	34.3	0.148	0.218	0.350
fft_a_md3	0.533	0.531	0.533	11.0	11.0	11.0	0.097	0.145	0.300
pci_bridge32_a_md1	0.655	0.652	0.653	45.8	45.7	45.7	0.227	0.329	1.217
pci_bridge32_a_md2	0.861	0.839	0.829	38.3	18.7	18.1	0.301	0.467	1.263
pci_bridge32_b_md1	0.796	0.781	0.778	54.2	51.4	51.4	6.718	0.309	0.447
pci_bridge32_b_md2	0.745	0.704	0.698	72.3	61.7	61.7	0.158	0.316	0.652
pci_bridge32_b_md3	0.966	0.925	0.879	49.8	49.8	49.8	0.171	0.336	0.713
Norm. Avg.	1.02	1.00	**1.00**	1.31	**1.00**	1.06	1.92	**1.00**	2.26

* Window expands horizontally (vertically) by four (two) rows for each iteration.
† Window expands horizontally (vertically) by six (four) rows for each iteration.
‡ Window expands horizontally (vertically) by eight (six) rows for each iteration.

J. Chen et al. (2017) by 8%, in total displacement.

3.3.3 Trade-off on Window Size

In the third experiment, we verify the trade-off between running time and solution quality on the expansion step of window size. As shown in Table 3.4, a larger window size is benifitial on averge displacement while a smaller window size causes 2% degradation. On the other hand, performing MGL takes less time with a smaller window in most of the designs, expect for dense regions where smaller windows need more expansion iterations.

Table 3.5: Running Time of Multi-Threaded MGL

Benchmark	MGL Running Time (s)			
	1T	2T	4T	8T
des_perf_1	15.66	9.16	6.12	4.74
des_perf_a_md1	9.61	5.29	2.97	1.81
des_perf_a_md2	8.52	4.74	2.62	1.67
des_perf_b_md1	6.06	3.44	2.04	1.28
des_perf_b_md2	7.01	3.99	2.19	1.31
edit_dist_1_md1	5.14	3.14	1.64	0.98
edit_dist_a_md2	7.35	4.23	2.36	1.30
edit_dist_a_md3	10.55	6.76	3.88	2.78
fft_2_md2	1.44	0.87	0.46	0.29
fft_a_md2	0.91	0.55	0.32	0.22
fft_a_md3	0.85	0.49	0.24	0.15
pci_bridge32_a_md1	1.63	1.08	0.55	0.33
pci_bridge32_a_md2	1.79	1.23	0.70	0.47
pci_bridge32_b_md1	1.15	0.80	0.46	0.31
pci_bridge32_b_md2	1.32	0.86	0.48	0.32
pci_bridge32_b_md3	1.38	0.88	0.48	0.34
Norm. Avg.	1.00	0.61	0.34	0.22

3.3.4 Efficiency of Multi-threading

In the fourth experiment, we verify the efficiency of multi-threaded MGL. Since the solutions are identical with different numbers of threads and multi-threading is only implemented for MGL, we only show the running time of MGL in Table 3.5, where $1.6\times$, $2.9\times$, and $4.5\times$ speedup is achieved with two, four, and eight threads, respectively. In practice, the window expands horizontally by six rows and vertically by four rows for each iteration.

□ End of chapter.

Chapter 4

Multi-row Cell Detailed Placement

Summary

In this chapter, we propose a bipartite-matching-based maximum displacement optimization and a network-flow-based average displacement optimizations. Evaluated with ICCAD 2017 Contest designs, these two detailed placement algorithms achieve 1% and 66% less average and maximum displacement respectively.

To tackle one of the most critical stages in the physical synthesis flow, placer not only needs to provide legal placement solution, but also need to consider various objectives including timing, congestion, and power, which are optimized in global placement. Given a legalized input, the duty of detailed placement is to restore the optimized objectives while preserving the legality constraints such as power and ground (P/G) alignments, fence region, and routability (e.g., edge spacing, pin short/inaccessible). In this chapter, we present fast and high-quality detailed placement algorithms for standard cells with mixed cell heights optimizing both the maximum and average displacement. The methods handling fence constraints and routability constraints (e.g.,

edge spacing and pin access/short) are also discussed. Our major contributions are as follows.

- Iterative bipartite graph matching is devised to minimize the maximum displacement among a group of cells that can exchange their positions without creating additional violations.
- We extend the min-cost flow (MCF) formulation of the fixed-row-and-order problem to one that optimizes a weighted sum of the maximum and average displacement, with range constraints on the cell movements to avoid pin short and pin access violations.

The rest of the chapter is organized as follows. Section 4.1 illustrates the problem formulation and constraints. Section 4.2 provides a detailed explanation of our proposed techniques. Section 4.3 verifies the effectiveness of our approach.

4.1 Problem Formulation

Given a set of m multi-row height cells $C = \{c_1, c_2, \cdots, c_m\}$. The cell height and displacement are measured in terms of the multiple of single row height. Let $\mathcal{H}$ be the set of cell height, $C_h \subseteq C$ be then set of cells whose height is h. The problem is to place each cell c_i from global placement (x'_i, y'_i) into (x_i, y_i), where $\{x_1, x_2, \cdots, x_m\}$ and $\{y_1, y_2, \cdots, y_m\}$ are sets of integers (i.e., cells are aligned to placement sites) and cells are overlap-free, such that the average displacement Eq. (3.2) and maximum displacement are minimized (Darav et al., 2017).

The power and ground (P/G) alignment (Chow et al., 2016) and the fence region constraint (Bustany et al., 2015) are treated as hard constraints. The routability constraints including edge spacing and pin short or inaccessible are considered as soft constraints (Yutsis et al., 2014).

Given a legalized placement, two special cases for average and maximum displacement optimization are formulated in this chapter. The first is to optimize for a specific type of cells. The second is to optimize with the y-coordinate and horizontal order of cells fixed.

4.2 Algorithms

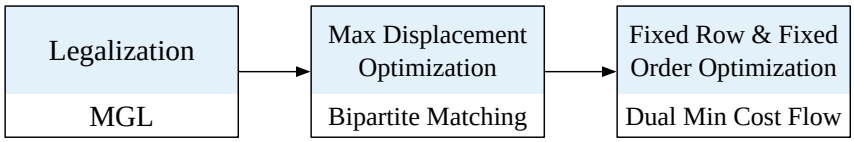


Figure 4.1: The proposed detailed placement flow.

The overall algorithmic flow is illustrated in Fig. 4.1, which consists of three stages.

1. Given a global placement (GP) solution with multi-cell-height circuits, we first legalize it by multi-row global legalization (MGL) which inserts the cells sequentially into the placement region.

2. Next, the maximum displacement is optimized by swapping cells of the same type in the same fence region. The displacement cost function is linear at the beginning and exponential afterward to maintain the average displacement at the same time.

3. Finally, keeping the rows and cell order unchanged, the average and maximum displacement is further optimized by linear programming.

Details of the last two steps are explained in the following sub-sections.

Algorithm 4.1 Maximum Displacement Optimization

Require: Cell list C.
Ensure: Legal positions of C.

while *true* **do**
  $c_t \leftarrow$ cell with max displacement;
  $C_T \leftarrow$ cell list with the same type and fence;
  if $|C_T| > s_0$ **then**
    Sort C_T by $\left|x_i - \frac{x_t+x'_t}{2}\right| + \left|y_i - \frac{y_t+y'_t}{2}\right|$;
    $C_T \leftarrow$ top-s of C_T;
    if $c_t \notin C_T$ **then**
      $C_T \leftarrow C_T \cup \{c_t\}$;
    end if
  end if
  Optimize C_T by bipartite matching;
  if max displacement of C_T is not changed **then**
    return ;
  end if
end while

4.2.1 Maximum Displacement Optimization

In this section, we will present a maximum displacement optimization method in a legal placement. Recall that in MGL, each cell is processed sequentially and it will then be fixed to a segment once placed. The maximum displacement can be further reduced if the row assignments can be changed, especially for the cells being placed near the end of MGL. It is unavoidable to place them with large displacements if the regions around their GP locations are dense. Figure 4.4(a) shows the displacement of a cell type in a fence region. Each rectangle represents a cell. Red cells are of the same type and gray cells are of other types. The long gray lines connect cells to their corresponding GP positions. We can see that some cells are placed to even tens of rows away from their GP positions.

To reduce the maximum displacement without creating violations in the legal placement, we perform a iterative min-cost bipartite matching to optimize the max-

imum displacement. Algorithm 4.1 shows the flow of the optimization. In each iteration, the target cell c_t with the maximum displacement is selected (line 2). Cell list $C_T \subseteq C$ is initialized by cells with the same type and fence region as c_t (line 3). If $|C_T|$ is larger than a given number s_0, only c_t and the cells close to $(\frac{x_t+x'_t}{2}, \frac{y_t+y'_t}{2})$, the middle point of the current and GP position of c_t, are selected for fast running time (lines 4 to 10). Given a bipartite graph $G = (C_T, \mathcal{P}_T, C_T \times \mathcal{P}_T)$ on the current positions $\mathcal{P}_T \subseteq \mathcal{P}$ of the cells in $\mathcal{P}_T$, any cell $c_i \in C_T$ can take up the positions $p_j = (x_j, y_j)$ of another cell c_j to minimize the maximum displacement without creating any violations. The problem is to find a perfect matching $\mathcal{S} \subseteq C_T \times \mathcal{P}_T$ between cells and positions with the minimum total cost $\sum_{(c_i,p_j)\in S} D_{i,j}$, where $D_{i,j} = \phi(|x_j - x'_i| + |y_j - y'_i|)$ and $\phi(\delta)$ is defined as a strictly increasing function such that it is linear when δ is small to preserve the average displacement. After a certain threshold of δ, ϕ will increase rapidly in order to discourage large displacement. Here, we have:

$$\phi(\delta) = \begin{cases} \delta, & \delta \leq \delta_0, \\ \frac{\delta^5}{\delta_0^4}, & \text{otherwise}, \end{cases} \tag{4.1}$$

where δ_0 is the tolerable maximum displacement threshold. This min-cost perfect matching problem can be optimally solved by formulating as a min-cost flow (MCF) problem (Király & Kovács, 2012).

There are previous works that use bipartite matching in detailed placement to minimize HPWL (G. Chen et al., 2018) but only those cells on "independent" nets can be optimized simultaneously. Here the cost function ϕ is defined in such a way that both the average and maximum displacement are handled and all selected cells of the same type can be optimized simultaneously.

4.2.2 Fixed-row and Fixed-order Optimization

After the matching-based maximum displacement optimization, we perform a final post-processing refinement to further reduce the maximum and average displacement by shifting the cells locally without changing the cell order and row assignments. Taking the objective of the total displacement as an example, given a set of m multi-row height cells $C = \{c_i\}$, the problem can be formulated as follows:

$$\min_{x_i} \sum_i n_i \delta_{xi} \tag{4.2}$$

$$\text{s.t.} \quad x_i + w_i \leq x_j, \qquad \forall (i,j) \in \mathcal{E}, \tag{4.2a}$$

$$l_i \leq x_i \leq r_i, \qquad \forall c_i \in C, \tag{4.2b}$$

where n_i is the weight on the x-displacement δ_{xi} of cell c_i, w_i is the width of c_i, l_i and r_i are the left and right boundary of the segment containing c_i, $\mathcal{E}$ is the set of neighboring pairs where $(i,j) \in \mathcal{E}$ if and only if c_i is the left neighbor of c_j on some rows. The left and right boundries of segments can be determined by core boundries, fence regions, or placement blockages.

Equation (4.2) can be converted to a dual min-cost flow (MCF) problem and effectively solved (Y. Lin et al., 2018; Vygen, 1998). Compared to the MCF formulation by Y. Lin et al. (2018), our transformation to MCF has three strengths.

- There are significantly fewer vertices in the flow network, which is more efficient.
- The maximum and average displacement are optimized simultaneously.
- Weight n_i is set according to Eq. (3.2) while it is ignored (i.e., all the same) in the formulation by Y. Lin et al. (2018).

To obtain the MCF formulation, we first split the x-displacement δ_{xi} in Eq. (4.2)

to a pair of variables x_i^- and x_i^+ and achieve a linear programming (LP) formulation

$$\max_{x_i, x_i^-, x_i^+} \sum_i n_i (x_i^- - x_i^+) \tag{4.3}$$

$$\text{s.t. } x_i^- \le x_i - x_i' \le x_i^+, \qquad \forall c_i \in C, \tag{4.3a}$$

$$x_i^- \le 0 \le x_i^+, \qquad \forall c_i \in C, \tag{4.3b}$$

$$x_i - x_j \le -w_i, \qquad \forall (i,j) \in \mathcal{E}, \tag{4.3c}$$

$$x_i \ge l_i, \qquad \forall c_i \in C_L, \tag{4.3d}$$

$$x_i \le r_i, \qquad \forall c_i \in C_R, \tag{4.3e}$$

where C_L is the set of left-most cells in at least one of the segments and C_R is the set of right-most cells in at least one of the segments. Let $\tilde{x}_0$ be the absolute position of the *origin*, then the absolute positions $\{\tilde{x}_i, \tilde{x}_i^-, \tilde{x}_i^+\}$ of $\{x_i, x_i^-, x_i^+\}$ are $\tilde{x}_i = x_i + \tilde{x}_0$, $\tilde{x}_i^- = x_i^- + \tilde{x}_0$, $\tilde{x}_i^+ = x_i^+ + \tilde{x}_0$. By substitute the absolute positions for the relative positions, we have

$$\max_{\tilde{x}_0, \tilde{x}_i, \tilde{x}_i^-, \tilde{x}_i^+} \sum_i n_i (\tilde{x}_i^- - \tilde{x}_i^+) \tag{4.4}$$

$$\text{s.t. } \tilde{x}_i^- \le \tilde{x}_i - x_i' \le \tilde{x}_i^+, \qquad \forall c_i \in C, \tag{4.4a}$$

$$\tilde{x}_i^- \le \tilde{x}_0 \le \tilde{x}_i^+, \qquad \forall c_i \in C, \tag{4.4b}$$

$$\tilde{x}_i - \tilde{x}_j \le -w_i, \qquad \forall (i,j) \in \mathcal{E}, \tag{4.4c}$$

$$\tilde{x}_i - \tilde{x}_0 \ge l_i, \qquad \forall c_i \in C_L, \tag{4.4d}$$

$$\tilde{x}_i - \tilde{x}_0 \le r_i, \qquad \forall c_i \in C_R, \tag{4.4e}$$

whose dual linear programming is a MCF problem:

$$\min_{f} \; Q = \sum_{i} \left(x'_i (f_i^+ - f_i^-) - l_i f_i^l + r_i f_i^r \right) - \sum_{(i,j)\in\mathcal{E}} w_i f_{ij} \tag{4.5}$$

$$\text{s.t. } F_i = f_i^+ - f_i^- + f_i^r - f_i^l + \sum_{j:(i,j)\in\mathcal{E}} f_{ij} - \sum_{k:(k,i)\in\mathcal{E}} f_{ki} = 0, \qquad \forall c_i \in C, \tag{4.5a}$$

$$F_0 = -\sum_{i} F_i = 0, \tag{4.5b}$$

$$0 \le f_i^+, f_i^- \le n_i, \qquad \forall c_i \in C, \tag{4.5c}$$

$$f_i^l, f_i^r \ge 0, \qquad \forall c_i \in C, \tag{4.5d}$$

$$f_i^l = 0, \qquad \forall c_i \in C - C_L, \tag{4.5e}$$

$$f_i^r = 0, \qquad \forall c_i \in C - C_R, \tag{4.5f}$$

$$f_{ij} \ge 0, \qquad \forall (i,j) \in \mathcal{E}, \tag{4.5g}$$

where each variable in Eq. (4.4) corresponds to a vertex in the graph while the constraints in Eq. (4.4) are illustrated by directed edges and dual variables $\{f\} = \{f_i^-\} \cup \{f_i^+\} \cup \{f_{ij}\} \cup \{f_i^l\} \cup \{f_i^r\}$. Note that each of the auxiliary vertices $\{v_i^-\}$, $\{v_i^+\}$ connects only two edges which can be combined to form one edge. Hence, they can be eliminated. Overall, this is a MCF problem with $m + 1$ vertices and $2m + |C_L| + |C_R| + |\mathcal{E}|$ edges, while the MCF in (Y. Lin et al., 2018) has $3m + 2$ vertices and $6m + |\mathcal{E}|$ edges. Our formulation is simpler and thus can be solved more efficiently.

4.2.3 Extension Considering Maximum Displacement

The formulation above optimizes the total displacement. To consider the maximum displacement, we further introduce a pair of auxiliary variables δ^-, δ^+ whose absolute values represent the largest displacement of the cells to the left and to the right of the corresponding GP position. Thus, we extend Eq. (4.4) to consider a weighted sum as

follows:

$$\max_{\delta^-,\delta^+,\tilde{x}_0,\tilde{x}_i,\tilde{x}_i^-,\tilde{x}_i^+} \quad n_0(\delta^- - \delta^+) + \sum_i n_i(\tilde{x}_i^- - \tilde{x}_i^+) \tag{4.6}$$

$$\text{s.t.} \quad \delta^- \le \tilde{x}_i - x_i' - \delta_{y_i}, \qquad \forall c_i \in C, \tag{4.6a}$$

$$\delta^- \le \tilde{x}_0 - \delta_{y_i}, \qquad \forall c_i \in C, \tag{4.6b}$$

$$\delta^+ \ge \tilde{x}_i - x_i' + \delta_{y_i}, \qquad \forall c_i \in C, \tag{4.6c}$$

$$\delta^+ \ge \tilde{x}_0 + \delta_{y_i}, \qquad \forall c_i \in C, \tag{4.6d}$$

$$\text{Eqs. (4.4a) to (4.4e)},$$

where n_0 can be tuned to balance the maximum and the average displacement in the objective function and δ_{y_i} is the y-displacement of cell c_i, which are constants since the row assignments will be preserved in this step. The dual LP is as follows:

$$\min_f \quad Q + (f^p + f^n)\max_i \delta_{y_i} + \sum_i \left(x_i'(f_i^p - f_i^n) - \delta_{y_i}(f_i^p + f_i^n)\right) \tag{4.7}$$

$$\text{s.t.} \quad F_i + f_i^p - f_i^n = 0, f_i^p, f_i^n \ge 0, \qquad \forall c_i \in C, \tag{4.7a}$$

$$F_0 + f^n - f^p = 0, 0 \le f^p, f^n \le n_0, \tag{4.7b}$$

$$f^p - \sum_i f_i^p = 0, \tag{4.7c}$$

$$f^n - \sum_i f_i^n = 0, \tag{4.7d}$$

$$\text{Eqs. (4.5c) to (4.5g)},$$

where f^p, f^n, $\{f_i^p\}$ and $\{f_i^n\}$ are auxiliary variables for handling the maximum displacement.

Figure 4.2 shows an example. Cells c_1 and c_2 are single-row while cell c_3 is double-row. The corresponding flow graph is shown in Fig. 4.3. Vertices v_z, v_n and v_p are auxiliary nodes while each of the other nodes represents a cell in Fig. 4.2. The solid

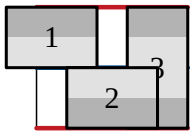


Figure 4.2: Global placement (GP) example.

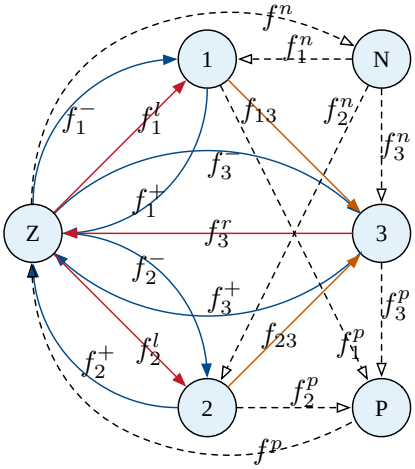


Figure 4.3: Flow network example.

straight edges from v_z (e.g., f_1^l) represent the flows for the constraints of the left boundary. The solid straight edge to v_z (there is only f_3^r in this toy case) represents the flow for the constraints of the right boundary. The other solid straight edges (e.g., f_{13}) illustrate the flows for the constraints between neighbouring cells. The solid curly edges (e.g., f_i^-) represent the flows formulating the absolute value. The dotted edges (e.g., f^n) represent the flows for the formulation of the maximum displacement. The capacity and cost of the edges are shown in Table 4.1.

In this work, we deploy a network simplex algorithm with first eligible pivot rule to solve the MCF problem. The worst case complexity is $O(nm^2QU)$ (Király & Kovács, 2012), where n and m denote the number of nodes and arcs in the flow network respectively, U denotes the maximum arc capacities and Q denotes the largest arc cost.

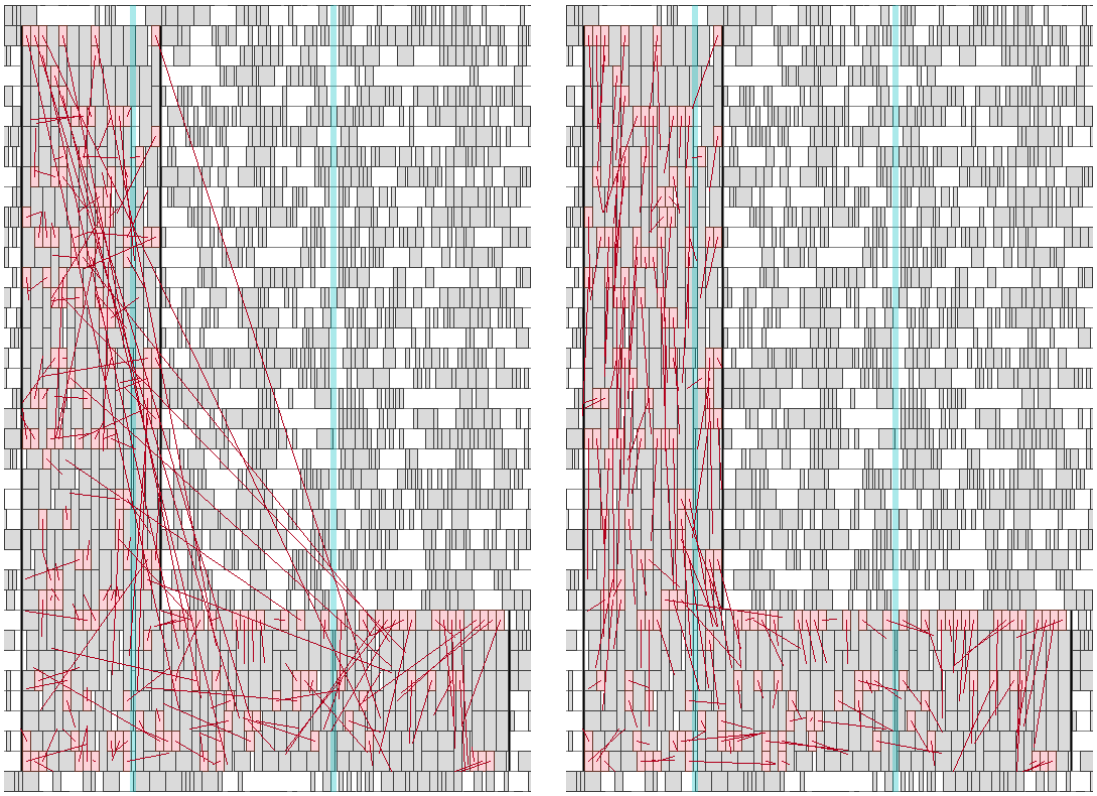
Table 4.1: Edge Capacity and Cost Example

Flow	Cap	Cost
f_1^-	1/4	1
f_1^+	1/4	-1
f_2^-	1/4	-1
f_2^+	1/4	1
f_3^-	1/2	-3
f_3^+	1/2	3
f^n	1/50	0
f^p	1/50	0
f_1^n	∞	1
f_1^p	∞	-1
f_2^n	∞	-1
f_2^p	∞	1
f_3^n	∞	-3
f_3^p	∞	3
f_{13}	∞	-3
f_{23}	∞	-3
f_1^l	∞	0
f_2^l	∞	0
f_3^r	∞	5

4.2.4 Routability-driven Refinement

Edge spacing rules define the minimum distances between different types of cells. The method in Sec. 4.2.1 does not create violations to any edge spacing rules because only cells of the same type replace each other in the bipartite matching and all pairs of consecutive cell edges remain unchanged. For the fixed row and fixed order optimization, fillers will be inserted for correct edge spacing when calculating the width of the cell on the left of the insertion point so constraint Eq. (4.4c) becomes

$$\tilde{x}_i - \tilde{x}_j \leq -w_i - e_{i,j}, \forall (i,j) \in \mathcal{E}, \tag{4.8}$$



(a) Before (b) After

Figure 4.4: Maximum displacement optimization.

where $e_{i,j}$ is the edge spacing requirement for cell pair (c_i, c_j).

Pin access and pin short violations are caused by signal pins of cells sheltered by P/G rails or I/O pins so that no same layer wire or upper layer via can access the pins without causing design rule violations. The method in Section 4.2.1 does not create pin access or pin short violations. Furthermore, to avoid more pin access and pin short violations in the fixed row and fixed order optimization, the cells will be restricted to a feasible range defined by the intersection of the row segment and the P/G rails or I/O pins. Thus, every cell has its left and right boundary constraints in the MCF, i.e., $C_L = C_R = C$. l_i and r_i in the formulation are the left and right boundaries of cell c_i's feasible range respectively.

Table 4.2: Results of Post-Processing

Benchmark	Avg. Disp.		Max. Disp.	
	Before	After	Before	After
des_perf_1	0.964	0.915	49.5	10.0
des_perf_a_md1	0.919	0.919	60.7	60.7
des_perf_a_md2	1.148	1.148	48.1	48.1
des_perf_b_md1	0.675	0.662	53.8	11.4
des_perf_b_md2	0.618	0.619	27.3	24.1
edit_dist_1_md1	0.664	0.664	5.8	5.7
edit_dist_a_md2	0.614	0.614	16.5	16.4
edit_dist_a_md3	0.783	0.762	74.2	23.3
fft_2_md2	0.721	0.721	10.0	6.5
fft_a_md2	0.563	0.563	34.3	34.3
fft_a_md3	0.531	0.531	11.0	11.0
pci_bridge32_a_md1	0.652	0.652	45.7	45.7
pci_bridge32_a_md2	0.839	0.834	18.7	18.1
pci_bridge32_b_md1	0.781	0.781	51.4	51.4
pci_bridge32_b_md2	0.704	0.704	61.7	61.7
pci_bridge32_b_md3	0.925	0.925	49.8	49.8
Norm. Avg.	1.01	**1.00**	1.66	**1.00**

4.3 Experimental Results

We implemented the proposed detailed placement algorithm for mixed-cell-height circuits in C++ programming language. LEMON (Király & Kovács, 2012) is used as the MCF solver. We run all experiments on a 64-bit Linux machine with eight cores of Intel Xeon 2.1GHz CPUs and 64GB RAM.

We verify the effectiveness of the two post-processing stages. Figure 4.4 shows an example of the maximum displacement optimization in which red cells are of the same type while red lines connect cells to their corresponding GP positions. Cells with large displacement in Fig. 4.4(a) are moved to closer locations in Fig. 4.4(b). Table 4.2 lists the average displacement and maximum displacement before and after the post-processing stages. We can see that through the proposed optimization, the maximum

displacement and the average displacement can be decreased by around 66% and 1%.

□ End of chapter.

Chapter 5

Scalable Detailed Routing

Summary

In physical design flow, detailed routing takes on the rule from detailed placement to fulfill complicated design rules and to access pins on cells with congested intra-cell routing. In this chapter, we propose a detailed router named Dr. CU 2.0 that judiciously handles hard-to-access pins and emerging design rules including length-dependent parallel-run-length spacing, end-of-line spacing with parallel edges, and corner-to-corner spacing. The experimental results illustrate that our framework can effectively reduce the number of design rule violations with comparable wirelength. Comparing Dr. CU 2.0 with the best score of each design in the ISPD 2019 Initial Detailed Routing Contest, there is 2% score improvement. Compared with the state-of-the-art (G. Chen, Pui, Li, Chen, et al., 2019), our algorithm achieves 69% better scores. The source code of Dr. CU 2.0 is available at github.com/cuhk-eda/dr-cu.

Separated into a global step and a detailed step, regular net routing is to produce metal wires running parallel to the x-axis or y-axis to create connections between pins of a net (Gester et al., 2013). Shrinking feature size and increasing design complexity introduce complicated design rules and thus enormous computation effort to routing. Global routing divides the die area into coarse-grain grids called G-cells to model the design rules by edge capacity and optimize routing resource usage and wire delay, crosstalk, power, etc. (G. Chen & Young, 2020). In some approaches, global routing is followed by track assignment to provide a suggestive wiring solution (Ding & Yildiz, 2020). Being extensively studied for more than five decades, detailed routing is decisive in VLSI design flow by carrying on the cause of electrical feasibility and manufacturability from detailed placement to minimize design rule violations (Kahng et al., 2020b). In this chapter, we propose a detailed routing algorithm handling pin access and design rules in advanced technology nodes. Our contributions can be summarized as follows.

- We design a pre-processing stage to compute the valid access points of each pin and create off-track vias if no same-layer access point is valid.
- We propose a design rule-aware maze routing to handle end-of-line spacing with parallel edges in a correct-by-construction manner and fix corner-to-corner spacing violations in post-processing.
- We develop a lookup-table-based via insertion method and select violation-free via types from the cell library.
- Our proposed method outperforms the state-of-the-art works including the champion of the latest detailed routing contest.

The rest of the chapter is organized as follows. Section 5.1 outlines the basic terminologies and new design rules handled in this work. In Sec. 5.2, we describe the proposed algorithms to improve pin access and to perform design-rule-aware routing, whose effectiveness is evaluated in Sec. 5.3.

5.1 Preliminaries

In this section, we will describe some basic terminologies and concepts related to the routing grid graph, design rules, and constraints.

5.1.1 Routing Grid Graph

Better routability can be achieved when wires with different directions are separated to different metal layers. For each metal layer, cell libraries define a *preferred direction* with which the routing wires are preferred to be aligned. Preferred directions of adjacent layers are usually perpendicular to each other. For example, a typical case is to avoid routing on the first metal layer, route horizontally on the second and fourth metal layers, while vertically on the third and fifth metal layers. *Tracks* on each metal layer are ideal routing positions along the preferred direction defined during floorplanning. Wires routed on tracks are less likely to incur violations since the tracks are well spaced according to the technology and constraints.

Because of the perpendicular nature between the preferred directions on adjacent layers, each track on a layer will intersect with the projections of the tracks on its adjacent layers at many points. All these intersection points are considered as grid points to form a 3D grid graph on which maze routing will be performed. For the edges in the grid graph, besides the edges connecting neighboring grid points on the same track, there are also cross-layer edges that connect grid points vertically aligned on adjacent layers. For efficient query and memory usage, we adopted a two-

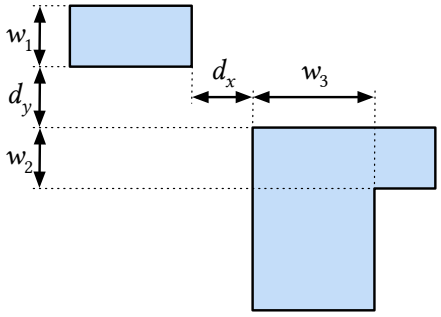


Figure 5.1: Corner-to-corner spacing.

level framework that consists of a global grid graph and local ones. The global grid graph maintains the information of all the routed wires and inserted vias. The local grid graph maintains the edge costs from the information on the global grid graph to perform maze routing for a net.

5.1.2 Design Rules and Constraints

In advanced technology nodes, resolution enhancement techniques (RETs) including optimized model-based optical proximity correction (MBOPC), phase-shifting masks, and off-axis illumination (OAI) with sub-resolution assist features (SRAFs) are applied in the post-layout design flow for optical lithography equipments to reduce the resolution limit beneath one-eighth of the light wavelength (Schellenberg, 2003). Lithographers and layout engineers defined set of design for manufacturability (DFM) rules to create layout patterns compliant with RETs (Jhaveri et al., 2010). In this section, we illustrate three new design rules highlighted in the recent ISPD 2019 Initial Detailed Routing Contest (W.-H. Liu et al., 2019).

MBOPC emphasizes corner stressing serifs to prevent energy scarcity for sufficient exposure (Otto et al., 1994). However, aggressive MBOPC will potentially cause metal shorts between close by corners. The corner-to-corner spacing rule requires a safe distance which increases monotonically with the maximum width of involved

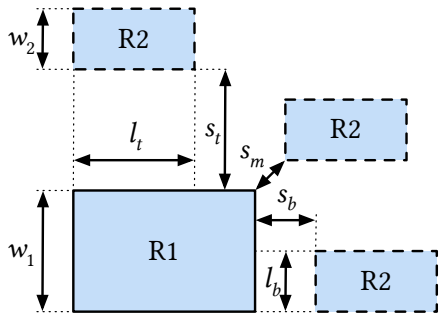


Figure 5.2: Parallel run length spacing.

corners. An example is shown in Fig. 5.1. If the widest rectangle of a metal corner is wider than a specified width $eolWidth$, and if there is another metal corner and their PRL is non-positive, i.e., they do not overlap in the x- or y-directions, then, their L_{inf} distance should not be less than the corner-to-corner (C2C) spacing which can be looked up from a table according to the value of $\max(w_1, w_2, w_3)$ for this example.

In sub-ninety-nanometer technologies, the parallel run length (PRL) spacing requirements between two rectangles of different nets depend on both width and PRL of the two rectangles (Qi et al., 2015). Considering the three cases of different relative positions between two metal rectangles (R1 and R2) whose top-down view is shown in Fig. 5.2, the effective width $\max(w_1, w_2)$ is the same among all the three cases but the PRLs are different. Longer PRL l_t results in more significant crosstalk effect and thus requires larger spacing s_t while shorter PRL l_b requires smaller spacing s_b. Rectangles that do not overlap in the x- or the y- direction are considered as having negative PRL and their spacing in Euclidean distance s_m should not be less than the zero PRL spacing requirement.

As another promising technique among computational RETs, multiple patterning lithography (MPL) that assigns the conflicted patterns to separated masks for manufacturing aims at aggressively cutting the resolution limit to one-fourteenth of the light wavelength (Chang, 2019). However, a metal end-of-line (EOL) with a parallel edge in an adjacent track causes additional stitches (Yu et al., 2015), which may lead

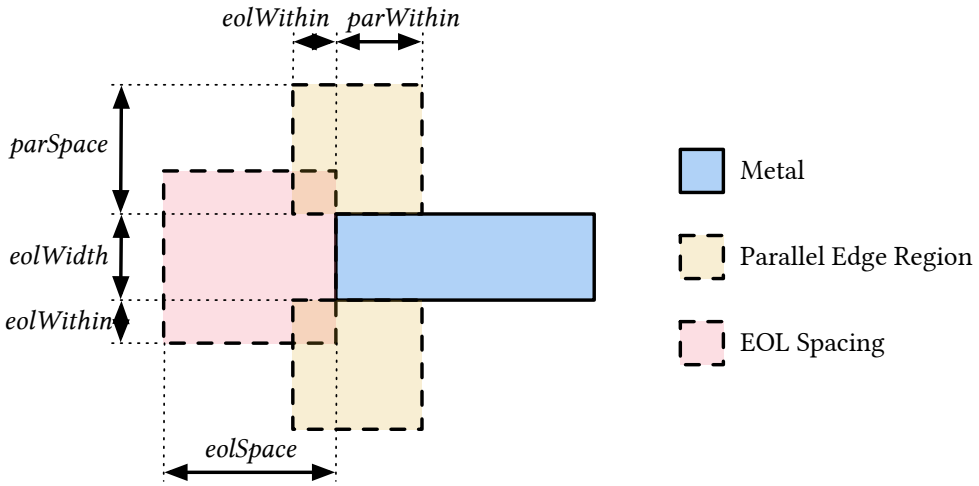


Figure 5.3: End-of-line spacing with parallel edge.

to yield degradation or even cause native conflicts that are indecomposable in MPL. A new kind of EOL spacing requirement that depends on the existence of other parallel edges is defined for the generation of lithography-friendly layouts. On both sides of the EOL, two yellow parallel edge regions are defined as shown in Fig. 5.3. Their width is the parallel edge spacing *parSpace* extending from the point at a distance of $eolWithin$ in front of the EOL to the point at a distance of $parWithin$ behind the EOL. This EOL spacing rule applies, i.e., no metal overlaps with the red region, if a parallel routing wire overlaps with the parallel edge regions.

5.2 Algorithms

The overall algorithmic flow of our router is illustrated in Fig. 5.4. It consists of three stages.

1. Given a placement solution with special net structures and a global routing solution in the form of route guides for the detailed router, we first assign access points for each cell pin and I/O pin, which will be explained in Sec. 5.2.1. Note that the special nets have been routed and their structure should not be

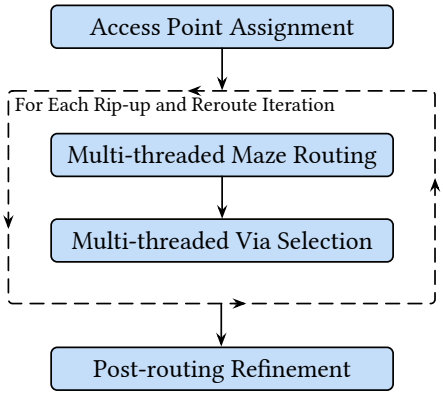


Figure 5.4: Proposed routing flow.

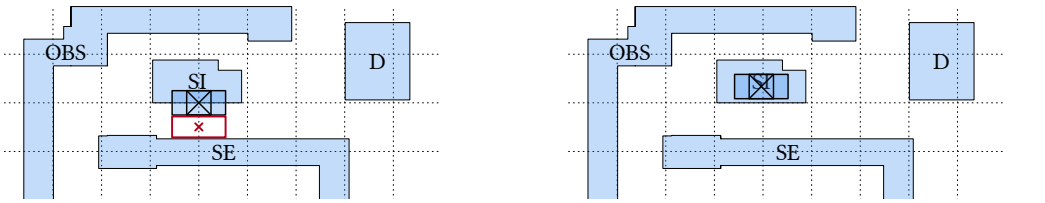
changed.

2. Next, the nets are routed one by one, followed by rounds of rip-up and reroute. In each round of rip-up and reroute, the route guides are expanded. Maze routing of each net is distributed to a multi-threading scheme, followed by the via type selection for each net, which is also multi-threaded, as described in Sec. 5.2.2.

3. Finally, a post-routing refinement stage is conducted to resolve C2C spacing violations. Some additional techniques to handle the new spacing constraints will be explained in Sec. 5.2.3.

5.2.1 Pin Access

Pin access is one of the most difficult problems in detailed routing. In order to simplify and tackle the pin access problem and to improve the routability, for each pin, we assign several same-layer grid points in its surrounding area as its *access points*. During the maze routing of the net connecting the pin, if one of the access points is reached by the routing path, the pin is considered as being connected. An L-shape metal will be inserted to finish the connection in case the access point reached is not



(a) Same-layer access point creates a violation. (b) Off-track via connects upper access points.

Figure 5.5: Off-track pin access.

sufficiently connected to the pin. We also penalize the grid points on the upper layer above the pin for other nets to guarantee that the pin can be reached and fade the penalty after the pin is successfully connected.

With the advancement of technology nodes, the number of routing tracks in a single placement row decreases. The limited number of in-cell tracks has brought high routing density and mutual blockages between accessing wires of neighboring pins. For some pins, there is even no violation-free same-layer grid point that can be assigned as an access point. More specifically, no via type can be used to connect to the pin directly or by a routing wire without causing any design rule violation. An example is shown in Fig. 5.5(a), where pin `SI` is embraced by other pins and a via inserted at the grid point marked with a black cross will violate the PRL spacing rule with pin `SE` below. In fact, there is no grid point at which a via can be inserted to connect to `SI`, as either the selected via or the routing wire will cause violations.

To handle this issue, we will also assign grid points on the upper layer as access points and use possibly off-track vias to connect pins with upper-layer access points. An example is shown in Fig. 5.5(b), where the grid points surrounding the pin location on the upper layer are marked as access points of the pin. A proper off-track position (e.g., the center point of the pin shape) is selected to insert a violation-free via to reach the upper-layer access point. These off-track vias to resolve pin access problems are identified before maze routing and will not be moved during rip-up and reroute. Therefore, these off-track vias are marked as obstacles during routing to

avoid violations with the wires and vias of other nets.

5.2.2 Via Type Selection

To enhance routability, multiple via types are provided to connect a pair of adjacent routing (or masterslice) layers through the cut layer in between. However, determining the via location and via type simultaneously is challenging due to the complicated metal shapes and cut patterns in the surrounding of the vias. To handle this efficiently, we will first construct a set of via conflict lookup tables (LUTs). Next, via insertion and via type selection will be performed for each net separately during routing. We will first determine the via locations and generate the routing topology for a net. Via type selection will then be performed to find the best via type locally. After routing all the nets, all via type and location information will be available, an additional round of via type selection will be performed in a post refinement stage to finally decide the best via types for the nets globally.

5.2.2.1 Construction of Conflict LUTs

The via insertion and via type selection processes require violation detection between the inserted via and the surrounding wires, vias, pins, and obstacles. These query operations happen so frequently that the run time of on-the-fly detection for every via insertion candidate position is not affordable. Fortunately, since we are working on a relatively regular grid graph, a set of light-weight LUTs can be constructed to accelerate the process. In general, via-related spacing violations can be divided into three categories:

1. Via-pin/obstacle conflicts: a via may have spacing violations with fixed pins or obstacles on the lower or the upper metal layers that it connects. The locations with this kind of violations will be marked before the routing stage and will be avoided during via insertion.

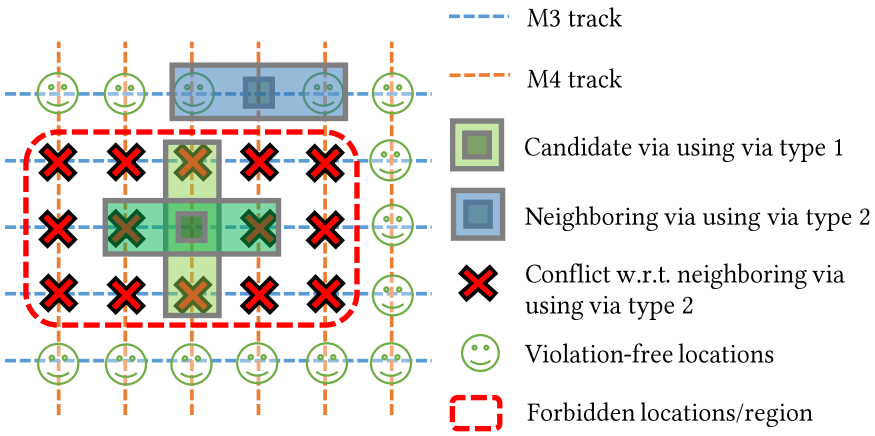


Figure 5.6: Via-via LUT.

2. Via-wire conflicts: a via may have spacing violations with the wire on the lower or the upper metal layers that it connects.

3. Via-via conflicts: a via may have spacing violations with not only the vias on the same cut layer but also with vias on adjacent cut layers.

For each via type, the conflict LUTs with wires on the lower or the upper metal layer are built to provide immediate responses for queries about whether or not a neighboring edge conflicts with a given via type at a given location. Similarly, for each pair of via types on the same cut layer or on adjacent cut layers, a conflict LUT is built. For example, for the pair of via type 1 and via type 2 in Fig. 5.6, a conflict LUT is built to mark the neighboring *forbidden regions* of via type 1 with respect to via type 2. When inserting a via of type 1 at the center, any routed via of type 2 inside the forbidden regions of via type 1 will trigger a conflict and we need to ensure that is not the case before inserting the type 1 via.

5.2.2.2 Via Type Selection with Pessimistic Query

When routing a net, we first assume that all the newly inserted vias are of the default via type, and determine their locations by minimizing the number of conflicts with the

surrounding wires, vias, pins, and obstacles. After generating the routing topology of the net, via type selection is performed to determine the best type at the given via location. Meanwhile, a *global routed via map* is built and maintained to record the locations and types of all the settled vias.

Obviously, for a given via type at a given location, the efficiency of detecting the surrounding via-via violations will dominate the run time of the entire selection process. This is because the surrounding vias may be of different types and we need to look up different via-via conflict LUTs. A naive implementation is, for a given via type, all its via-via LUTs are checked one by one to count the accumulated violation number with respect to its surrounding vias. However, a lot of redundancy will be introduced with such an approach, since each location may be visited several times, and a long run time will be resulted.

In order to avoid repeated queries on the surrounding locations, an alternative approach is to identify all those neighboring vias that may conflict with the given via type using one single via-via LUT. Conceptually, this approach is conducted in a pessimistic manner, that is, a neighboring via (regardless of its via type) will first be marked as a suspicious conflicting via if it is inside *any* forbidden region of the given via type at the given location. After that, all suspicious conflict vias will be checked by looking up into their respective via-via LUTs to verify whether there are real conflicts.

Based on this idea, we construct a set of pre-computed *merged via-via LUTs* to find out all the neighbouring suspicious conflict vias of a given via type in a one-time-effort. As illustrated in Fig. 5.7, for a specific via type (e.g. via type 1 at the centre), the corresponding merged via-via LUT marks the union of the forbidden regions of via type 1 with respect to all other possible via types (e.g. via types 1, 2 and 3). Any neighbouring via (regardless of its via type) inside the forbidden region union of the merged LUT will be marked as a suspicious conflicting via. In the next verifying

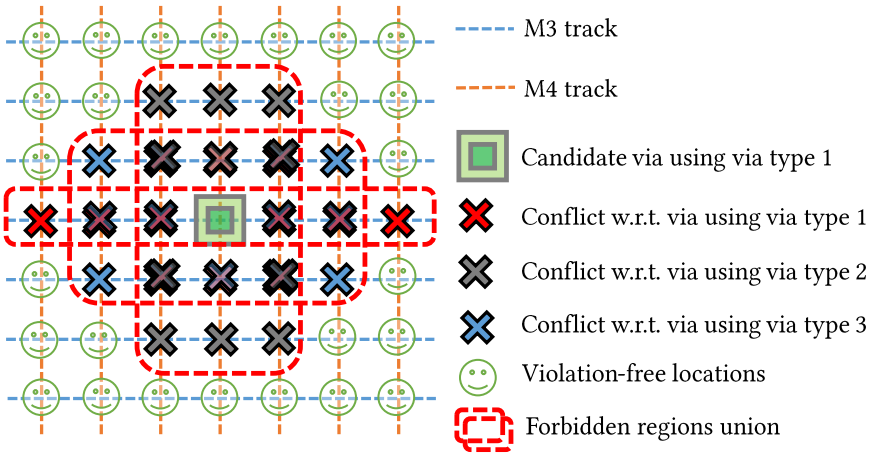


Figure 5.7: Merged via-via LUT.

stage, if a neighbouring suspicious via has already had its type selected, its via type can immediately be looked up from the global routed via map (default via type is used otherwise). We can then identify if a neighbouring suspicious via is truly conflicting with the given via type at the given location by accessing the respective via-via LUTs. As a result, the best via type with the lowest violation count can be selected efficiently.

5.2.2.3 Multi-stage Selection and Multi-thread Scheduling

Via type selection is an order-sensitive process. Different orders of via type selection may lead to different results, and an optimal selection result may require simultaneous type changes for adjacent vias. Thus, with the information provided by the global routed via map, an additional round of via type selection is performed in the post refinement stage to reduce the order sensitivity of the selection results. We have implemented an efficient multi-thread scheduling scheme for via type selection, which can significantly accelerate the multi-stage via type selection process. Nets without routing topology overlap are assigned to different batches, and the via type selections of different batches will be executed in parallel.

5.2.3 More Techniques for New Design Rules

Techniques for handling corner-to-corner (C2C) spacing and end-of-line (EOL) spacing with parallel edges introduced in the ISPD 2019 Initial Detailed Routing Contest are presented as follows.

5.2.3.1 Corner-to-corner Spacing

There are three sources for the corner-to-corner (C2C) spacing violations described in Sec. 5.1.2, i.e., the violations can be caused by vias, wires, or pins/obstacles. We only handle the first two, vias and wires, as the majority of pins are on Metal 1 but there are rarely C2C rules on the bottom layer. For the violations caused by vias, we handle the C2C spacing rule using the LUTs described in Sec. 5.2.2. For the violations caused by wires, they only exist at the corners of wrong-way wires because based on our observation, the $eolWidth$ for C2C spacing is wider than the wire width. Therefore, we will transform the corners of wrong-way wires to EOLs by adding small metal rectangles on track and handle the EOL spacing in maze routing.

5.2.3.2 End-of-line Spacing

There are two kinds of end-of-line (EOL) spacing constraints on each metal layer: with or without parallel edges. On each layer, the *eolSpace* (see Sec. 5.1.2) of the end-of-line constraint with parallel edges (EWP) is larger than that without parallel edges (EOP). When a metal violates only the EWP (not the EOP) but the EWP does not apply since there is no parallel edge overlapping with the parallel edge regions, the track segments within the parallel edge regions should be forbidden to use in order not to trigger the EWP violation. In this case, some available tracks will be blocked in the middle, which can affect routability. To handle this issue, we only consider the EOP but with the spacing requirement *eolSpace* increased to that of the EWP. This pessimistic method is practical since, in most of the cases, the two *eolSpace* values

are close to each other and it is not beneficial to handle the complex conditions for the EWP.

5.3 Experimental Results

We implement the proposed detailed routing algorithm in C++ programming language. Rsyn (Flach et al., 2017) is used as the parser of the LEF/DEF file format. We run all experiments on a 64-bit Linux machine with eight cores of Intel Xeon 2.2 GHz CPUs and 64 GB RAM.

5.3.1 Comparison with Contestants

In the first experiment, we compare our proposed algorithm with the best score of each design in ISPD 2019 Initial Detailed Routing Contest. The basic characteristic of the designs and the experimental results are listed in Table 5.1. The running time of the best scores among contestants is measured on the machines of contest organizer as a reference so that running time cannot be compared directly. Our proposed algorithm performs 2% better on average. The weights of the cost function used in the contest and the detailed results obtained by our algorithm is listed in Table 5.2. The units used in the wirelength and short area calculation are the Metal 2 pitch and the square of the Metal 2 pitch, respectively. We can see from the table that only marginal percentages of the wires and vias are routed in a non-preferred way.

5.3.2 Comparison with the State-of-the-art

In the second experiment, we compare our initial detailed router with the state-of-the-art (G. Chen, Pui, Li, Chen, et al., 2019) and list the results in Table 5.3. Note that the router by G. Chen, Pui, Li, Chen, et al. (2019) does not handle the new design rules

Table 5.1: Comparison with The Best Scores in ISPD'19 Contest.

Design	# Std. Cells	# Block Macros	# Nets	# I/O Pins	# Layers	Die Size (mm^2)	Tech. Node	ISPD'19 Quality Score		Time (s)	
								Best	Ours	Best	Ours
ispd19_test01	8879	0	3153	0	9	0.148×0.146	32 nm	626129	631347	150	260
ispd19_test02	72094	4	72410	1211	9	0.873×0.589	32 nm	21942353	21460183	1442	2491
ispd19_test03	8283	4	8953	57	9	0.195×0.195	32 nm	1044187	1060117	63	89
ispd19_test04	146442	7	151612	4802	5	1.604×1.554	65 nm	22755592	21370286	1622	2877
ispd19_test05	28920	6	29416	360	5	0.906×0.906	65 nm	3301288	3292562	157	269
ispd19_test06	179881	16	179863	1211	9	1.358×1.325	32 nm	47916252	46130808	2918	4108
ispd19_test07	359746	16	358720	2216	9	1.581×1.517	32 nm	99164297	98256678	7174	9458
ispd19_test08	539611	16	537577	3221	9	1.803×1.708	32 nm	137211940	136114789	10335	13102
ispd19_test09	899341	16	895253	3221	9	2.006×2.151	28 nm	216753037	212577797	14173	17987
ispd19_test10	899404	16	895253	3221	9	2.006×2.151	32 nm	216333270	212777964	14955	18673
Avg. Ratio								1.02	1.00	0.66	1.00

Table 5.2: Detailed Metrics on ISPD'19 Designs.

Design	Basic Use		Non-preferred Use					Rule Violations							
	WL‡	# V‡	Out-of-guide WL	Out-of-guide # V	Off-track WL	Off-track # V	Wrong-way WL	# Shorts	Short Area	# Min Area	# PRL	# EOL	# Cut	# Adj‡	# C2C‡
Weight	0.5	4.0	1.0	1.0	0.5	1.0	1.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0
ispd19_test01	642383	36889	12499	1617	767	741	10884	96	36	33	9	23	15	3	58
ispd19_test02	24943322	808060	363055	33552	21884	23633	207317	1323	450	556	870	831	198	88	5920
ispd19_test03	842146	65566	12259	1667	1962	661	16081	74	25	45	209	119	33	76	109
ispd19_test04	30484232	1033993	568497	48221	15201	2	122225	1481	670	36	260	44	0	0	0
ispd19_test05	4779913	153447	13530	2699	2090	18	15245	279	68	3	150	13	0	0	0
ispd19_test06	66019435	1991004	662101	64728	25138	12507	444711	2898	1008	656	1096	689	296	49	1229
ispd19_test07	122505150	4821351	855312	101530	48185	24324	608986	1915	601	1780	21580	1556	830	47	3900
ispd19_test08	188414168	7350652	1203856	162298	76715	36396	731245	2480	693	2585	4091	2649	1594	86	6488
ispd19_test09	285335800	12226079	2052346	276625	118145	60675	1215030	4989	1599	4913	6591	4696	2202	127	9567
ispd19_test10	282121356	12514819	2088315	256880	119526	60656	1418869	5273	1823	4705	6646	4374	1964	116	10646

‡ WL denotes wirelength. # V, # Adj, and # C2C denote the numbers of vias, adjacent cut and corner-to-corner violations.

Table 5.3: Comparison with the State-of-the-art on ISPD'18 Designs.

	Design	Basic Use		Non-preferred Use					Rule Violations			ISPD'18 Quality Score	Mem (GB)	Time (s)
		WL	# V	Out-of-guide WL	Out-of-guide # V	Off-track WL	Off-track # V	Wrong-way WL	Short Area	# Min Area	# Spc			
	Weight	0.5	2.0	1.0	1.0	0.5	1.0	1.0	500.0	500.0	500.0	–	–	–
Ours	ispd18_test01	433538	32406	2187	442	414	0	5838	0	0	2	291307	0.41	18
	ispd18_test02	7832669	325618	40852	6534	5294	0	54151	0	0	57	4700255	1.86	141
	ispd18_test03	8718434	318353	68284	6673	6002	0	60028	379	0	97	5371855	3.84	377
	ispd18_test04	26410849	727732	347634	28839	17560	0	202113	86	93	588	15631883	12.36	1997
	ispd18_test05	27800973	965450	145035	21927	5841	3	76977	62	138	358	16357191	9.88	1109
	ispd18_test06	35702778	1480379	243634	36493	16548	16	118972	12	251	547	21624661	6.57	984
	ispd18_test07	65171367	2402251	395533	55611	33447	0	186904	145	364	186	38392522	14.40	2067
	ispd18_test08	65467807	2411866	393637	54455	33486	0	183358	156	390	177	38567127	15.12	1985
	ispd18_test09	54758969	2410569	357192	55216	26461	0	177411	10	513	113	33121862	11.23	1287
	ispd18_test10	68098610	2597471	1107400	101283	43250	0	235592	1206	595	736	41978776	11.91	3557
	Avg. ratio	1.00	1.00	1.00	1.00	1.00	–	1.00	–	–	1.00	1.00	1.00	1.00
ASPDAC‡	ispd18_test01	434914	34443	4352	859	276	0	2363	15	0	122	362725	0.32	17
	ispd18_test02	7817285	339055	104720	11784	4353	0	22023	1330	0	1949	6366886	1.15	121
	ispd18_test03	8707641	331958	176736	10731	4344	0	22187	1982	0	2419	7430092	1.25	139
	ispd18_test04	26042785	701994	769265	31444	41791	0	89537	26329	0	11224	34112928	2.89	494
	ispd18_test05	27852167	942588	649224	43071	13390	0	63397	4722	0	7742	22805761	3.87	767
	ispd18_test06	35813473	1446807	976672	68656	20357	0	95811	12891	0	11023	33908653	5.16	1155
	ispd18_test07	65360688	2349580	2187794	101866	33105	0	170316	33041	0	14880	63816462	8.86	2071
	ispd18_test08	65668468	2360231	2288159	102982	33373	0	170583	22353	0	14384	58501486	8.92	2060
	ispd18_test09	54993356	2358857	1604576	115465	29620	0	168722	17316	0	14470	50010785	8.52	2016
	ispd18_test10	68282001	2532666	2826908	140343	32865	0	180586	150705	0	20837	128141527	8.98	2132
	Avg. ratio	1.00	1.00	3.62	1.75	1.20	–	0.68	–	–	49.86	1.69	0.59	0.85

‡ G. Chen, Pui, Li, Chen, et al. (2019).

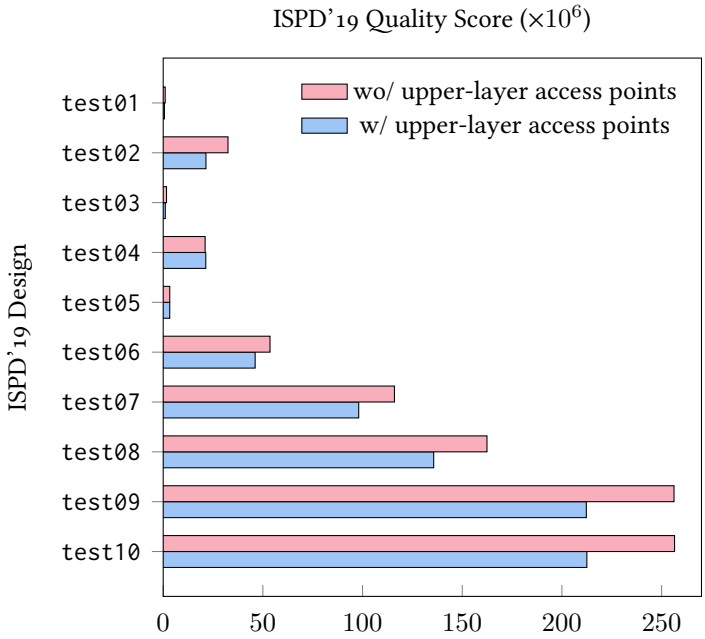


Figure 5.8: ISPD'19 quality score with different settings.

like the length-dependent PRL spacing, EOL spacing with parallel edges, and corner-to-corner spacing. Therefore, we compare our router with theirs using the designs of ISPD 2018 Intial Detailed Routing Contest (Mantik et al., 2018). The column # Spc denotes the total number of spacing violations as the evaluator in the contest does not provide the detailed number of violations on each design rule. Our proposed algorithm significantly improves the quality score by 69% with 41% more memory consumption and comparable running time. The route guide is better honored by our detailed router as the out-of-guide wirelength is reduced by more than 3 times. At the same time, we resolved almost all of the short area. The number of spacing rule violations is reduced by more than 49 times.

5.3.3 Comparison with Different Settings

In the third experiment, we verify the effectiveness of off-track pin access described in Sec. 5.2.1. As shown in Fig. 5.8, with upper-layer access points, the average score

improves by 26%.

□ End of chapter.

Chapter 6

Split Manufacturing Attack

Summary

To close timing with sufficiently short wirelength, significant effort is conducted by physical design tools — cells from the same net are placed close to each other and are connected by shallow and light wires — which, unfortunately, discloses back-end-of-line (BEOL) connections in front-end-of-line (FEOL). In this chapter, we challenge the security promise of split manufacturing by formulating various layout-level placement and routing hints in FEOL as vector-based and image-based features. We construct a sophisticated deep neural network which infers the missing BEOL connections with high accuracy. Compared with the network-flow attack (Y. Wang et al., 2018), we achieve on average a $1.21\times$ correct connection rate (CCR, the higher the better) when splitting after M1 and a $1.12\times$ CCR when splitting after M3, with less than 1% running time. Compared with (Zeng et al., 2019), ours incurs 47% of the size of the candidate list (the smaller the better) with only a 1% loss on accuracy.

Accomplishing the physical design on state-of-the-art technology nodes, fabless design houses have to rely on outsourced, offshore foundries for cost-effective access to cutting-edge fabrication, which enables various attack avenues on intellectual property (IP) for the external foundries. The notion of integrated circuit split manufacturing, which delegates the front-end-of-line (FEOL) and back-end-of-line (BEOL) parts to different foundries, is to prevent overproduction, intellectual property (IP) piracy, and targeted insertion of hardware Trojans, all by adversaries in the FEOL facility. When doing so, the untrusted foundries cannot get control of the full design while the fabless design houses can still benefit from the access to the latest technology node.

In this chapter, we leverage deep learning to thoroughly learn the characteristics of chip layouts arising from their physical design. This notion of learning is fundamentally more powerful than prior art, as any deviations for the notion of proximity (e.g., due to heavy congestion) can be learned on from the layouts. To the best of our knowledge, this is the first DL-based attack on split manufacturing that provides better results than the state-of-the-art, non-learning-based attacks (Y. Wang et al., 2018). Our methods also resolve the imbalance problem encountered by another learning-based attack (Zeng et al., 2019) which has to use the same number of negative and positive samples. The major contributions of our work are summarized as follows:

- We leverage deep learning for attacking split manufacturing. Using TensorFlow 2.1, we design and train a sophisticated deep neural network (DNN) architecture, which can predict the missing BEOL connections for an unknown FEOL layout with high accuracy.

- Our neural network makes use of vector-based and image-based layout features simultaneously. The feature structure is compatible with a wide range of

designs while saving memory and time consumption.

- The proposed *softmax regression loss* allows our attack to directly and effectively select the most probable BEOL connection among the relevant candidates without suffering from the imbalance between positive and negative samples as traditional classifiers.

The rest of this chapter is organized as follows. Section 6.1 outlines the threat model and problem formulation. Section 6.2 contains our features for the deep learning attack. In Sec. 6.3, we illustrate the architecture and configuration of the proposed DNN. The effectiveness of both attack and defense are verified in Sec. 6.4.

6.1 Preliminaries

6.1.1 Threat Model

Consistent with prior work (Y. Wang et al., 2018; Zeng et al., 2019), we assume that attackers have access to the full design information of the FEOL layers. Hence, they can identify the gates and pins, the related FEOL routing, and the resulting but incomplete netlist. Also, they know the maximum load capacitances from the cell library and can estimate an upper bound for the delay. Further, consistent with prior work, we assume attacks take place while chips are being fabricated. Hence, oracle access is not available for the attacker. We acknowledge S. Chen and Vemuri (2018), where an oracle was leveraged to assist during the attack on split manufacturing, but here we adopt the classical, stronger threat model where the chip is either yet-to-be manufactured or is not available at all in the market. We also assume that an attacker has a database of layouts generated in a similar manner as the one under attack.

The objective of an attacker residing in the untrusted FEOL foundry is to decipher the missing BEOL interconnects solely from the available FEOL information. The

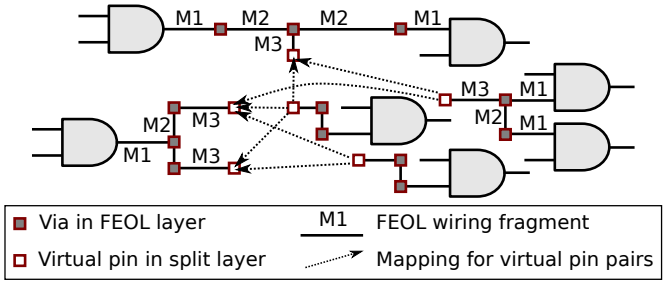


Figure 6.1: Terms for learning on split manufacturing layouts. Examples of virtual pin pairs are shown by dashed arrows pointing from sink fragments to source fragments.

corresponding goal is to reconstruct the design and ultimately to pirate the chip IP, overproduce the chip, or insert hardware Trojans.

6.1.2 Problem Formulation

Split layer refers to the top-most FEOL layer, while *virtual pins* are vias manufactured to connect the FEOL with the BEOL (Zeng et al., 2019). During split manufacturing, *fragments* are connected parts of FEOL wires, holding at least one virtual pin in the split layer. There are two different types of fragments as shown in Fig. 6.1:

- *Source fragment*: a driver/source along with fragments which are routed up until and within the split layer;
- *Sink fragment*: a fragment which is routed within the split layer and down towards the sink pin(s). For multi-fanout nets, the sink pins may be routed together in the FEOL as one sink fragment or separately as several sink fragments.

Given (as in extracted from the FEOL) are a set of m sink fragments, each of which has $c_1, c_2, \ldots, c_m$ sink pins, and a set of source fragments. *Virtual pin pairs* (VPPs) are mappings between virtual pins in sink fragments and virtual pins in source fragments. A VPP which is truly connected in the BEOL is called a *positive VPP*. Otherwise, it is

called a *negative VPP*. The connection predicting problem is to select a VPP for each sink fragment maximizing the correct connection rate (CCR) which can be thought of as the percentage of sink pins that are successfully restored (Y. Wang et al., 2018).

We define CCR as follows:

$$CCR = \frac{\sum_{i=1}^{m} c_i x_i}{\sum_{i=1}^{m} c_i}, \tag{6.1}$$

where $x_i = 1$ (respectively, 0) when a positive (respectively, negative) VPP is selected for the i-th sink fragment. Note that sink pins which do not belong to any sink fragment are excluded from consideration by definition, as this part of the IP is already fully exposed in the FEOL. Therefore, it is important to keep in mind that CCR serves well as a measure for attack effectiveness, but not so much for IP protection.

6.2 Feature Extraction

The BEOL is only available at training time, where the true connectivity is extracted to label VPPs as positive or negative ones. The FEOL is available for both training and testing/attacking phases. Hence, all features have to be tailored for and extracted from the FEOL. We propose two feature categories for our deep learning attack, vector-based and image-based features, and we explain how to integrate these heterogeneous features into a unified DNN architecture in Sec. 6.3.

6.2.1 Vector-based Features

6.2.1.1 Distances for VPPs

The working essence of computer-aided design (CAD) tools inspires the distance features, as placers typically place connected gates closer to each other (H. Li et al., 2018) and detailed routers typically route wires along the shallowest and lightest violation-

free paths (H. Li, Chen, et al., 2019). Still, by the virtues of (a) being able to learn on various layouts and (b) the joint working of these and all other features proposed in this work, any deviations from this essence can be captured as well. This is because the set of features have been devised to represent a physical layout in reasonable detail while remaining agnostic to particular design characteristics.

In accordance with routing principles, the distances for VPPs arising along the preferred and non-preferred routing directions are considered separately. To mitigate scaling issues across different layouts used for the same model, instead of measuring distances by database unit, distances are normalized by the pitch of the metal tracks in the split layer. All distances are also duplicated and normalized separately, by encoding in the ratios of the chip width or height, respectively. Therefore, designs based on different technology nodes and exhibiting different floorplan shapes and dimensions are made compatible for joint training and testing/attack.

6.2.1.2 Number of Sink Pins and Load Capacitance

These features track the total load value and the number of sink pins for each VPP. Instead of estimating the driver strength by cell area Zeng et al. (2019), we directly capture the capacitance from the cell library which is available to the attacker. As we are handling split or incomplete layouts, the load capacitances can only be defined by two bounds, namely

- upper bound: maximum capacitance of the driver;
- lower bound: capacitance of the sink pins connected within the sink fragment, plus wire capacitances of the two related source and sink fragments.

While the upper bound is derived with regard to the source fragment and the lower bound is derived with regard to the sink fragment, their relationship hints the possiblility of BEOL connection. A higher upper bound represents stronger capability of

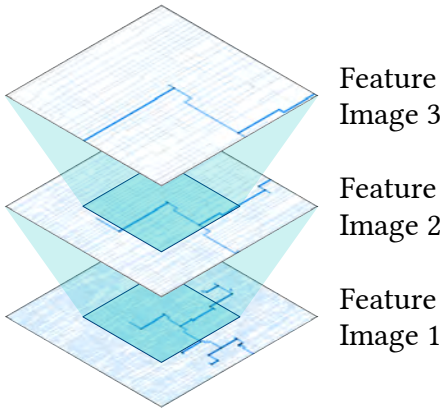


Figure 6.2: Layout image scaling.

driving sink fragments but too high upper bound may consume extra power.

6.2.1.3 FEOL Layer Wirelengths and Vias

These features capture the wirelength contribution in each FEOL metal layer individually. Contributions are tracked separately for the two fragments of a VPP. Within each layer, all wire paths of a fragment are summed up. The number of vias in each FEOL cut layer is also considered.

6.2.1.4 Driver Delay

For each VPP, we track the driver delay based on the underlying timing paths. Note that timing paths obtained from split layouts can only provide lower bounds for delays, as the paths may be incomplete. Thus, this feature tends to become more meaningful for higher split layers, when more of the paths are already completed in the FEOL.

6.2.2 Image-based Features

For each virtual pin, we represent the routing around the virtual pin as gray-scale layout images. Hence, the routing information of a VPP is represented in two sets

of layout images. To capture possible detours in routing, we consider three different scales with the same image shape but different precisions as shown in Fig. 6.2. Each image is 99 pixels wide and high, representing 99×99 consecutive regions. Each region is one, two, and four pitches of the split layer wide and high for the finest, medium, and the coarsest image, respectively.

There are two properties of the routed wires which will be encoded in the layout images of a virtual pin: the nets they belong to and the layers they are routed on. Let m be the number of metal layers in the FEOL. The total number of bits in a pixel to represent the layout information is $2m$, and we call these bits *layer bits*. The $2m$ layer bits are needed because wires of the same fragment as the virtual pin and wires from all other fragments are to be represented by different layer bits; the first m most-significant bits represent the routed wires of the virtual pin's fragment while the remaining m least-significant bits represent the wires of other fragments. Since wires closer to the BEOL carry more information about the connection, those in higher metal layers are encoded in more significant bits while those in lower metal layers are encoded in less significant bits. Vias connecting two layers are represented in both layer bits. More specifically, a '1' is assigned to the b-th bit with $b = m, \ldots, 2m - 1$ in a pixel if the virtual pin's fragment is routed in metal layer $b - m + 1$ in that region. Similarly, a '1' is assigned to the b-th bit with $b = 0, \ldots, m - 1$ in a pixel if there is some wire or via arising from other fragments in metal layer $b + 1$ of that region.

Figure 6.3 shows part of the image data which is representative for a part of the layout as in Fig. 6.1, where the split layer is M3, so that there may be wires on three different layers. Routed wires in the six consecutive regions bounded by the dashed lines are encoded into 2×3 pixels. Note that here we only show the values of the sixth, fifth, and fourth layer bits at the corner of each region, which together represent an exemplary virtual pin's fragment.

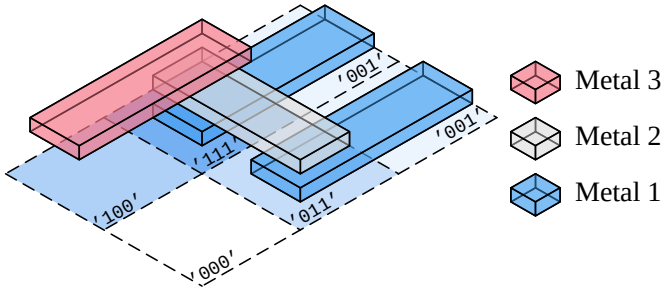


Figure 6.3: Layout image representation.

Feature extraction for large designs with, e.g., more than a million fragments will be time-consuming. Considering a fragment f, constructing a layout image of fragments other than f means to check which of all these other fragments hold wires in the nearby regions. Noticing that the information carried by a layout image of fragment f and the image of fragments other than f is equivalent to the information carried by the layout image of f and the layout image of *all* fragments, to manage the computational efforts, we let the m least-significant bits represent the wires of *all* fragments instead of *other* fragments, i.e., a '1' is assigned to the b-th layer bit when the $b+m$-th bit is '1' where $b = 0, \ldots, m-1$. Thus, to construct the image-based features in an efficient manner, at the beginning we construct a large layout image for all nets covering the whole die area and, thereafter, when generating a layout image centering any particular net, we only need to crop that large layout image to save most of the computational efforts incurred otherwise. Besides, for feature extraction running using t threads, t large layout images are constructed simultaneously and then merged together.

6.3 Deep Learning Framework

In this section, we first describe VPP selection for data cleaning. We then elaborate the DNN architecture and discuss our proposed SoftMax regression loss and its ad-

vantages.

6.3.1 Sample Selection

Due to an underlying tendency towards imbalanced datasets and long inference runtime, it is not practical to consider all possible VPPs, mainly because the correct connections are very few among all possible ones, which leads to a biased or inaccurate ML model. For N nets, even in the simplest scenario where (*i*) each FEOL wiring fragment holds only one virtual pin in the split layer and (*ii*) each net is split into exactly one source and one sink fragment, the sampling size is already N^2, whereas only $\frac{1}{N}$ samples are true positives.

Thus, based on three criteria discussed next, we select the n most relevant candidate VPPs for each sink fragment (irrespective of the number of sink pins in the fragment).

The first is the direction criterion. We apply a looser criterion than that proposed by Y. Wang et al. (2018) to avoid neglecting some positive VPPs, based on our observation that wires with non-preferred routing direction are relatively common in congested designs. For a VPP (p, q), where p is a virtual pin located at (x_p, y_p) and q is a virtual pin located at (x_q, y_q), if there is a horizontal or vertical wire segment connects p with a routing node p' located at (x'_p, y'_p), and q satisfies

$$\begin{cases} (x_q - x_p)(x'_p - x_p) \leq 0, & y'_p = y_p, \\ (y_q - y_p)(y'_p - y_p) \leq 0, & x'_p = x_p, \end{cases} \tag{6.2}$$

we then say the virtual pin p does *not rule out* virtual pin q, meaning that the two related fragments might be connected in the BEOL. More specifically, if p connects with no wire segment, which usually happens when p is a cell pin or I/O pin, we suppose $p' = p$, i.e, p does not rule out any virtual pin. Our direction criterion is

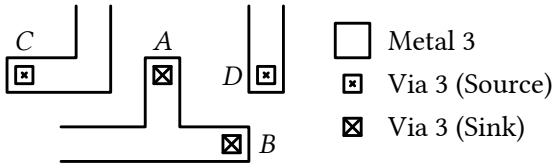


Figure 6.4: Examples for the direction criterion. Except VPP (B, C), all other VPPs are considered as candidates.

that a VPP is *not* considered as a candidate if and only if the above condition is *not met individually for both of the virtual pins.* In other words, a VPP is only disregarded if we find that neither the source fragment might be connected to the sink fragment nor vice versa. As indicated, this is a rather loose criterion, and particularly helpful to avoid neglecting some positive VPPs where parts of the related fragments are routed along non-preferred directions. Note that in case multiple virtual pins are present within a fragment, the condition is to be evaluated separately for each virtual pin. Also note that the final outcome of the direction criterion is independent of the order between virtual pins p and q; the criterion is symmetric.

The VPPs in Fig. 6.4 are evaluated in Table 6.1. For example, the wire of the source fragment connecting to the virtual pin C is pointing from right to left, while the virtual pin A of the sink fragment resides further to the right of C, so the condition in Eq. (6.2) is not met and we cannot say that the source fragment might be connected to the sink fragment. For the counterpart evaluation, required to decide on the criterion, note that the wire of the sink fragmenting connecting to A is pointing upward, while the virtual pin C is on the same height as A (i.e., C is just not below A), and the condition is met. Therefore, the direction criterion is fulfilled, and VPP (A, C) is still considered as a candidate.

The second criterion is relevance. If the sink and source fragments have multiple virtual pins, only the VPP(s) with the shortest distance apart in the routing direction orthogonal to the preferred direction of the split layer is (are) considered as candidate(s). This is based on the domain knowledge that net wirelengths are restricted

Table 6.1: Direction Criterion for VPPs in Fig. 6.4

Virtual Pin p / Virtual Pin q	A / C	A / D	B / C	B / D
p does not rule out q	✓	✓	✗	✓
q does not rule out p	✗	✓	✗	✓
Direction Criterion	✓	✓	✗	✓

to meet timing closure. It is important to keep in mind that metal stacks typically exhibit an alternating order for routing preferences. For example, consider the preferred routing direction in the split layer is horizontal, then the preferred direction for the next layer above the split layer—which is the first layer of the BEOL—is vertical. Based on the same domain knowledge, we assume that this first layer of the BEOL plays a significant role for the remaining wiring and, thus, the shortest distances in its preferred direction are leveraged in this criterion.

The third criterion is distance itself. If the number of VPPs remaining after considering the relevance criterion is still greater than n, the VPPs with shorter distance in the preferred direction of the first BEOL layer have a higher priority to be selected. Furthermore, if multiple VPPs are tied, the distance in the non-preferred routing direction is considered as a tie-breaker for the selection.

6.3.2 Model Architecture

For a sink fragment, we select n VPP of this sink and n different source fragments based on three criteria described in Sec. 6.3.1. We feed the neural network with input data constructed by this batch of n VPPs including the vector-based features of n selected VPPs, the image-based features of n source fragments in the related VPPs, and the image-based features of the sink fragment itself. The output data are scores for every VPPs in the batch. To handle vector- and image-based features in the same network, the proposed neural network illustrated in Fig. 6.5 first extracts underlying

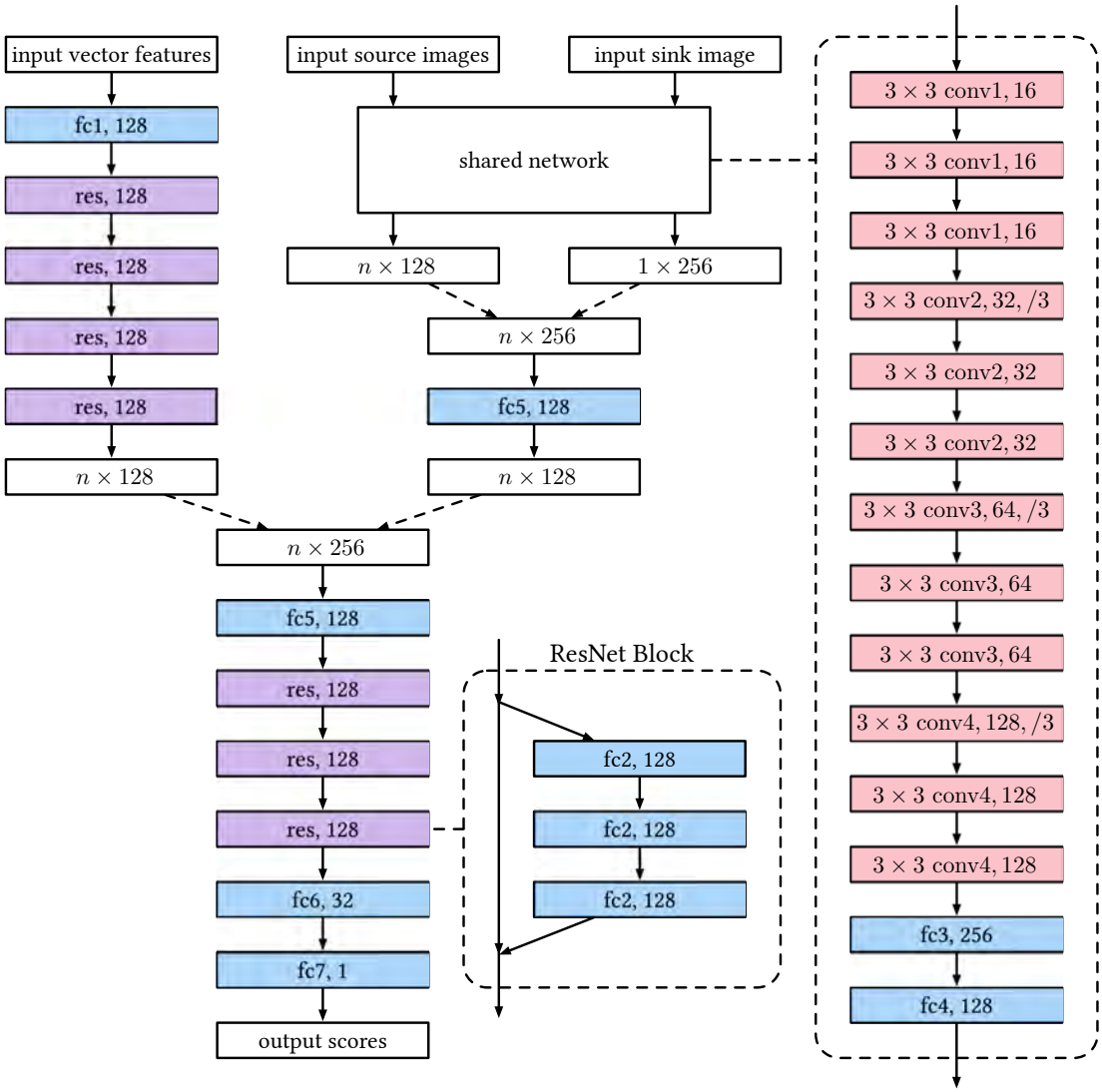


Figure 6.5: Convolutional neural network architecture.

features from heterogeneous input by processing vector-based features (shown in the upper left) and image-based features (shown in the upper middle) individually, and then processing them together (shown in the lower left) after concatenating the output of the vector and image part together.

For the image part of the network, note that the image-based features of the sink fragment are the same in the batch, so we only process them once, to save runtime,

Table 6.2: Convolutional Neural Network Configuration

Part	Layer	Parameter	Output
Vector part	fc1	27×128	$n \times 128$
	fc2	$[128 \times 128] \times 12$	$n \times 128$
Image part	conv1	$[3 \times 3, 16] \times 3$	$(n+1) \times 99 \times 99 \times 16$
	conv2	$[3 \times 3, 32] \times 3$	$(n+1) \times 33 \times 33 \times 32$
	conv3	$[3 \times 3, 64] \times 3$	$(n+1) \times 11 \times 11 \times 64$
	conv4	$[3 \times 3, 128] \times 3$	$(n+1) \times 4 \times 4 \times 128$
	fc3	128×256	$(n+1) \times 256$
	fc4	256×128	$(n+1) \times 128$
	fc5	256×128	$n \times 128$
Merged part	fc5	256×128	$n \times 128$
	fc2	$[128 \times 128] \times 9$	$n \times 128$
	fc6	128×32	$n \times 32$
	fc7	32×1	$n \times 1$

and its output is distributed to the output of every source images. Besides, all the image-based features go through the same shared network because the same set of information is needed to be extracted. Thus, each image-based feature is first processed individually through a shared convolutional neural network to reduce running time. Processing image-based features from source fragments and sink fragments through the same network can also make better use of all layout images. The shared network contains twelve convolution layers (red colored, labeled as `conv`) and two fully connected layers (blue colored, labeled as `fc`). The output from the sink image is then concatenated with every output from the source images and the combination passes through one more 128-way fully connected layer. For the vector part of the network, vector-based features are first transformed by a 128-way fully connected layer. Then, there are four residual networks (ResNet) blocks (purple colored, labeled as `res`) which can resolve the gradient vanishing problem while training very deep neural networks (He et al., 2016). The output of a ResNet block is the sum of its input and the output of three fully connected layers as shown in the middle sub-figure

of Fig. 6.5. After that, the output from the image part is concatenated with the output from the vector-based features. There is one 128-way fully connected layer to downsize the combination. The network ends with three ResNet blocks and two more fully connected layers. The filter and parameter configuration of the neural network is listed in Table 6.2. Both fully connected layers and convolutional layers are followed by a leaky rectified linear unit (LReLU) $y = \max(0.01x, x)$ as activation, where x is the input and y is the output (Maas et al., 2013).

6.3.3 Softmax Regression Loss

Given a query of a batch of n VPPs with at most one positive VPP, the network predicts the connection probability $s_1, s_2, \ldots, s_n$ for each VPP. The connection predicting task is to determine the index of the correct VPP to be connected:

$$\arg\max_i s_i, \tag{6.3}$$

as there can only be one source in a net.

While prior work handles similar problems as multi-class classification or two-class classification, e.g., see Zeng et al. (2019), we note that conventional multi-class classification approaches are in lack of two important properties for our work. In fact, conducting data augmentation requires exponential efforts if we were to use conventional multi-class classification methods. Firstly, the classification result for prior approaches depends on the order of classes, whereas for this work, the connection prediction should be independent of the order. Secondly, and more importantly, none of the prior methods can handle a variable number of classes, which is natural for the VPP connection prediction in our work, as this prediction is subject to a variable number of candidates.

In addition, simply modeling the VPP connection problem as a two-class classi-

fication problem is not appropriate, either. The main difference between our connection predicting problem and the classical regression problems is that we only care about the relative predicted probability between the only one positive VPP and the remaining negative ones, instead of their absolute values. Consequently, only the VPP with the largest predicted probability matters in the result. Moreover, an outlying negative VPP prediction would easily mislead the matching. Assuming a traditional two-class classification formulation, where the input of the neural network contains n VPPs with the same sink fragment, the loss of the two-class classification is

$$l_r = -\frac{1}{n}\left(\log \frac{e^{s_t^+}}{e^{s_t^-} + e^{s_t^+}} + \sum_{j \neq t} \log \frac{e^{s_j^-}}{e^{s_j^-} + e^{s_j^+}}\right), \tag{6.4}$$

whose partial derivative with respect to each score of either class is

$$\frac{\partial l_r}{\partial s_j^+} = -\frac{\partial l_r}{\partial s_j^-} = \begin{cases} -\frac{e^{s_j^-}}{n\left(e^{s_j^-} + e^{s_j^+}\right)} & \text{if } j = t, \\ \frac{e^{s_j^+}}{n\left(e^{s_j^-} + e^{s_j^+}\right)} & \text{otherwise,} \end{cases} \tag{6.5}$$

where s_j^+ and s_j^- are the scores of connection and non-connection for the j-th source fragment with $1 \leq j \leq n$ and t is the index of the true connection. The partial derivative with respect to the i-th weight of either neuron in the last fully connected layer is

$$\frac{\partial l_r}{\partial w_i^+} = -\frac{\partial l_r}{\partial w_i^-} = \frac{1}{n}\left(\sum_{j=1}^{n} \frac{e^{s_j^+} x_{i,j}}{e^{s_j^-} + e^{s_j^+}} - x_{i,t}\right), \tag{6.6}$$

where $x_{i,j}$ is the i-th input value of the last fully connected layer for the j-th source fragment. Therefore, the score of each source fragment acts independently on the gradient. The coefficient of the positive part of the gradient, which is due to the negative samples, is limited to 1 so that the VPP with even the largest connection probability will not dominate the gradient. As a result, misprediction of one VPP, which would

significantly influence our desired output as in Eq. (6.3), barely affects the average loss. Additionally, the numbers of positive and negative VPPs are imbalanced as most of the VPPs are negative samples. The negative part of the gradient, which is due to the only positive sample, is divided by the number of VPPs in the batch. Therefore, such a two-class classification model has a serious imbalance problem as it can easily gain a high accuracy by simply classifying all VPPs as negative, which is meaningless.

To resolve these problems, we consider only one score s_j for the j-th source fragment with $1 \leq j \leq n$. We propose the following *SoftMax regression loss*

$$l_c = -\log \frac{e^{s_t}}{\sum_{j=1}^{n} e^{s_j}}, \tag{6.7}$$

whose partial derivative with respect to each score of connection is

$$\frac{\partial l_c}{\partial s_j} = \begin{cases} \frac{e^{s_j}}{\sum_{j=1}^{n} e^{s_j}} - 1 & \text{if } j = t, \\ \frac{e^{s_j}}{\sum_{j=1}^{n} e^{s_j}} & \text{otherwise.} \end{cases} \tag{6.8}$$

The partial derivative of our proposed loss with respect to the i-th weight of the only neuron in the last fully connected layer is

$$\frac{\partial l_c}{\partial w_i} = \frac{\sum_{j=1}^{n} e^{s_j} x_{i,j}}{\sum_{j=1}^{n} e^{s_j}} - x_{i,t}, \tag{6.9}$$

in which the shortcomings of conventional two-class and multi-class classification model are resolved as follows. Firstly, the source fragment with higher score contributes more significantly in the gradient with an exponential factor. Let j_{max} be the index of the largest s_j. As the positive part of the loss is dominated by $x_{i,j_{max}}$, we have $\frac{\partial l_c}{\partial w_i} \approx x_{i,j_{max}} - x_{i,t}$. Secondly, the summation of the coefficients in the positive part equals to that of the negative part, so there is no imbalance issue. Thirdly, given any permutation of source fragments, the most probable source fragment is consistently

selected. Fourthly, the network can handle any number of source fragments as input. With these four advantages considered, the proposed SoftMax regression loss better reflects our way of computing the output as in Eq. (6.3), which is also supported by the empirical results.

6.4 Experimental Investigation

We conduct two sets of experiments as follows. In the first set, we evaluate the effectiveness of our proposed deep learning attack and compare it with the state-of-the-art network-flow attack (Y. Wang et al., 2018) and the machine-learning attack (Zeng et al., 2019). In the second set, we compare the performance of our attack against the network-flow attack for a particular congested design.

6.4.1 Evaluation and Comparison with State-of-the-art

6.4.1.1 Setup

In the first set of experiments, we derive a total of nine training and five validation designs from the ISCAS-85 (Hansen et al., 1999), MCNC (Yang, 1991), and ITC-99 benchmark suite (Corno et al., 2000). Concerning testing layouts, we use the same benchmarks as mentioned by Y. Wang et al. (2018) to ensure a fair comparison. We convert all designs to be combinational by ABC (Brayton & Mishchenko, 2010) and guarantee that the training, validation, and testing layouts are derived from different designs.

We use the academic NanGate $45\ \mathrm{nm}$ Open Cell Library (Knudsen, 2008) with ten metal layers for all of our experiments, also for those in subsequent sections. Synopsys Design Compiler M-2016.12-SP2 is used for synthesis and Cadence Innovus 17.1 is used for placement and routing. All training, validation, and testing layouts are devoid of any DRC violations. Once a layout is generated, we export the DEF file

and split the layout after the first or third metal layer, respectively, providing two sets to evaluate the attacks for different split layers.

We implement our feature extraction with C++ and train the model with Python and TensorFlow (Abadi et al., 2016). Without loss of generality, we select 31 VPPs for each sink fragment as the input of our deep learning attack based on the proposed criteria in Sec. 6.3.1. The learning rate is set as 0.001 and decayed to 60% for every 10 epochs. We execute all deep learning experiments on a 64-bit Linux machine with Intel Xeon 2.2 GHz CPUs and an NVIDIA Titan V GPU. We set the maximum runtime as a hundred thousand seconds (more than 24 hours) for all attacks and report CCR Eq. (6.1) as main metric. Recall that CCR serves well as a measure for attack effectiveness, but not so much for IP protection. Besides, all network-flow attacks are executed on a high-performance computing (HPC) facility where each computational node has two 14-core Intel Broadwell processors (Xeon E5-2680) running at 2.4 GHz. Further, each node has 128 GB RAM in total and 4 GB RAM are guaranteed for each attack by the Slurm HPC scheduler.

6.4.1.2 Results

In Table 6.3, we list the CCR for our proposed attacking method and the state-of-the-art (Y. Wang et al., 2018). We evaluate the success of the network-flow attack ourselves using the binary released in Feng et al. (2018). We note that the runtime of Feng et al. (2018) exceeds the limit on several large designs due to repetitive trials for removal of combinational loops. Our deep learning attack outperforms the state-of-the-art attack by $1.21\times$ and $1.12\times$ CCR when splitting after M1 and M3, respectively. Our inference time with feature extraction included is significantly shorter, namely only $0.001\times$.

We also compare our deep-learning-based attack with a traditional machine learning method based on random forest (Zeng et al., 2019). Originally, this attack pro-

Table 6.3: Comparison With (Y. Wang et al., 2018) on Selected ISCAS-85 and ITC-99 Benchmarks

Design	Split Layer: Metal 1						Split Layer: Metal 3					
	# Pins		CCR (%)		Runtime (s)		# Pins		CCR (%)		Runtime (s)	
	Sink	Source	TVLSI*	Ours	TVLSI*	Ours	Sink	Source	TVLSI*	Ours	TVLSI*	Ours
b07_C	520	235	8.43	10.19	326.13	8.55	115	51	55.65	84.35	0.67	3.62
b11_C	738	296	9.05	10.03	1719.46	11.06	213	57	66.67	66.67	0.94	4.20
b13_C	430	215	10.42	17.91	130.82	7.53	88	52	42.05	70.45	0.44	3.55
b14_C	6338	2864	N/A	8.57	100000+	77.62	2117	583	30.33	30.42	2576.42	16.08
b15_C	10176	3847	N/A	5.79	100000+	130.30	4910	1235	26.42	24.24	38292.53	33.50
b17_C	32385	12479	N/A	4.08	100000+	599.47	16190	4590	N/A	19.03	100000+	157.61
b18_C	84292	33703	N/A	4.59	100000+	2861.27	32719	9359	N/A	23.74	100000+	453.66
c1355	403	226	9.90	12.41	151.22	7.65	77	32	89.61	97.40	0.50	3.53
c1908	432	213	8.49	11.11	260.50	7.45	54	27	94.44	87.04	0.47	3.34
c2670	803	428	6.32	9.46	2251.82	11.70	206	120	54.85	58.74	1.48	4.64
c3540	1354	512	6.41	8.49	39187.25	17.55	452	124	54.87	51.11	7.39	5.42
c432	231	121	11.26	8.23	15.62	5.29	43	21	76.74	86.05	0.37	3.35
c5315	1919	847	7.50	9.33	94281.90	23.59	590	248	52.20	62.03	26.11	6.81
c6288	4124	2160	N/A	14.52	100000+	49.64	551	78	63.16	61.52	7.13	4.22
c7552	2008	1108	12.10	11.11	48656.51	22.82	296	175	50.34	72.30	7.64	3.72
c880	460	234	11.09	13.91	568.99	6.31	77	37	71.43	76.62	0.74	2.34
Average†			9.18%	11.11%	13889.37 s	10.67 s			59.20%	66.35%	2923.06 s	7.02 s
Ratio			1.00x	**1.21x**	1.000x	**0.001x**			1.00x	**1.12x**	1.000x	**0.002x**

* Y. Wang et al. (2018).

† For fairness, designs on which Y. Wang et al. (2018) times out are excluded for the calculation of average values.

Table 6.4: Comparison with Zeng et al. (2019) on Selected Benchmarks

Design	Accuracy (%)		$\|LoC\|$		Precision (%)		CCR (%)	
	TVLSI*	Ours	TVLSI*	Ours	TVLSI*	Ours	TVLSI*	Ours
b07_C	94.12	96.08	15.96	12.27	5.90	7.83	35.29	52.94
b11_C	82.46	85.96	23.60	20.60	3.49	4.17	21.05	64.91
b13_C	100.00	98.08	18.44	14.42	5.42	6.80	28.85	44.23
b14_C	85.93	76.67	103.11	47.39	0.83	1.62	15.61	43.74
b15_C	83.56	78.06	178.85	86.15	0.47	0.91	8.18	33.52
b17_C	61.39	58.98	273.02	127.05	0.22	0.46	4.81	22.33
b18_C	54.91	50.25	209.84	74.45	0.26	0.67	4.01	23.29
c1355	100.00	100.00	11.88	10.22	8.42	9.79	40.63	78.13
c1908	100.00	100.00	12.15	10.30	8.23	9.71	29.63	81.48
c2670	95.83	97.50	31.16	29.95	3.08	3.26	31.67	53.33
c3540	91.13	89.52	42.18	29.39	2.16	3.05	11.29	64.52
c432	95.24	95.24	7.10	11.10	13.42	8.58	52.38	85.71
c5315	95.56	96.77	65.77	47.25	1.45	2.05	21.37	52.42
c6288	82.05	82.05	29.77	8.72	2.76	9.41	35.90	78.21
c7552	98.86	99.43	41.51	28.34	2.38	3.51	32.57	53.14
c880	100.00	100.00	14.22	10.49	7.03	9.54	32.43	62.16
Average	88.82%	87.79%	67.41	35.51	4.10%	5.08%	25.35%	55.88%
Ratio	**1.00x**	0.99x	1.00x	**0.53x**	1.00x	**1.24x**	1.00x	**2.20x**

* Zeng et al. (2019).

vides only a list of candidates (LoC) for every fragment, no matter whether it is a source or sink fragment; we modify the code provided by Zeng et al. (2019) to only report the LoCs for sink fragments since an attacker can readily distinguish sink fragments from source fragments, which is relevant as an attacker needs to select a source for each sink fragment. We consider the three metrics proposed by Zeng et al. (2019), where $|LoC|$ designates the average size of the identified list of candidates for each testing benchmark, classification *accuracy* measures the number of times that the actual match of a fragment is included in its LoC, and *success rate of proximity attack*, which is identical to CCR. We introduce a fourth metric, called *precision*, which is the fraction of actual matching among LoC, calculated as accuracy over $|LoC|$.

Table 6.4 provides the results for (Zeng et al., 2019) and for our proposed attack. For ours, note that we select every VPP into the LoC whose score is higher than a reference value s_0; $s_0 = -8$ across all benchmarks. While achieving an almost identical accuracy, our $|LoC|$ is on average just $0.53\times$ of that of (Zeng et al., 2019), meaning that we can correctly infer the actual match using much smaller LoCs. In fact, we achieve on average $1.24\times$ the precision and even $2.20\times$ the CCR.

6.4.2 Ablation Studies

6.4.2.1 Setup

In the second set of experiments, we verify the effectiveness of our proposed SoftMax regression loss and the image-based features. For these experiments, we split the layout after the third metal layer and the baseline uses only the vector-based features with the loss Eq. (6.4) for simple two-class classification. The artificial neural network architecture using only the vector-based features is illustrated in Fig. 6.6.

6.4.2.2 Results

As shown in Fig. 6.7, the average CCR with the SoftMax regression loss Eq. (6.7) is $1.07\times$ that of the baseline. When additionally employing the image-based features, the average CCR further improves to $1.09\times$. We note that using the SoftMax regression loss also marginally improves the runtime. Thanks to the efficient layout encoding and network structure, the runtime when further using the image-based features is comparable to that of only using the vector-based features.

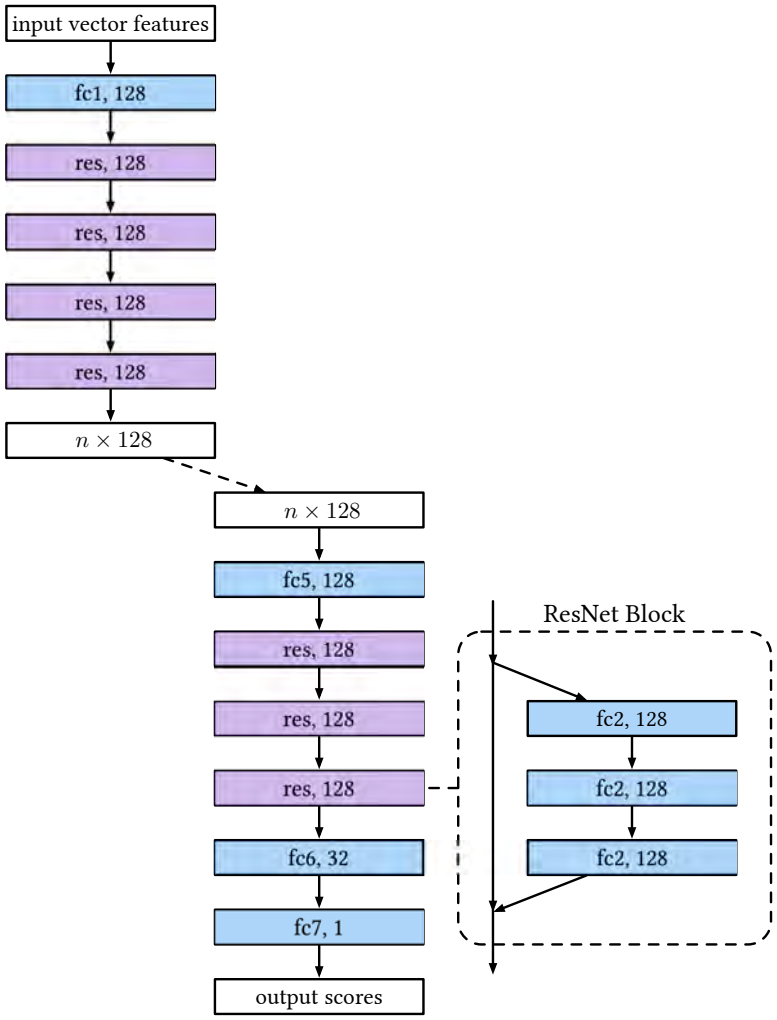


Figure 6.6: Artificial neural network architecture.

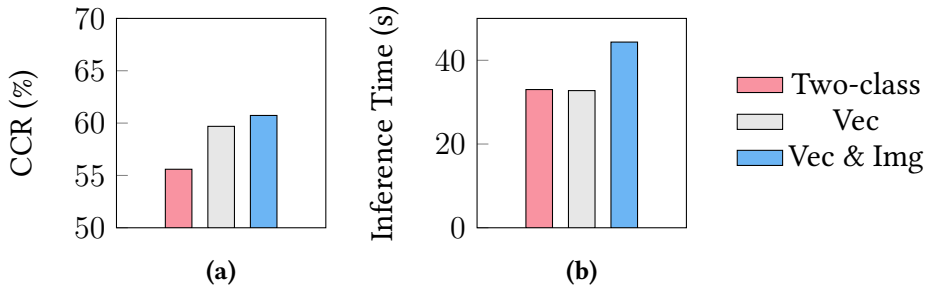


Figure 6.7: Comparison between different sets of features used in our method.

Table 6.5: Comparison between Knudsen (2008) and Vashishtha and Clark (2019)

Design	NanGate†			ASAP‡		
	# Pins		CCR (%)	# Pins		CCR (%)
	Source	Sink		Source	Sink	
b07_C	51	115	84.35	194	475	33.47
b11_C	57	213	66.67	265	664	35.09
b13_C	52	88	70.45	202	404	49.26
b11_C	57	213	66.67	265	664	35.09
b13_C	52	88	70.45	202	404	49.26
b14_C	583	2117	30.42	2283	5862	31.70
b15_C	1235	4910	24.24	3550	10002	25.09
b17_C	4590	16190	19.03	12443	34875	24.37
b18_C	9359	32719	23.74	22236	63208	30.00
c432	21	43	86.05	129	235	53.62
c880	37	77	76.62	206	424	43.16
c1355	32	77	97.40	176	332	47.29
c1908	27	54	87.04	212	420	49.76
c2670	120	206	58.74	375	697	40.46
c3540	124	452	51.11	448	1225	32.65
c5315	248	590	62.03	695	1632	32.48
c6288	78	551	61.52	1322	2787	62.83
c7552	175	296	72.30	793	1582	43.99

† NanGate 45 nm cell library (Knudsen, 2008).
‡ ASAP 7 nm cell library (Vashishtha & Clark, 2019).

6.4.3 Evaluation on Advanced Node Designs

6.4.3.1 Setup

In the third set of experiments, we derive a total of nine training and five validation designs from the ISCAS-85 (Hansen et al., 1999), MCNC (Yang, 1991), and ITC-99 benchmark suite (Corno et al., 2000). Concerning testing layouts, we use the same benchmarks as mentioned in (Y. Wang et al., 2018), to ensure a fair comparison. We guarantee that the training, validation, and testing layouts are derived from different designs.

We have thus conducted the first-ever attack on split manufacturing in the context of the 7 nm node, using the ASAP7 library (Vashishtha & Clark, 2019) with seven metal layers and the ISCAS-85 (Hansen et al., 1999) and ITC-99 (Corno et al., 2000) benchmark suites. Toward this end, Cadence Genus 17.1 is used for synthesis and Innovus 18.1 is used for placement and routing. All training, validation, and testing layouts are devoid of any DRC violations. Once a layout is generated, we export the DEF file and split the layout after the third metal layer. We present the CCR results in Table 6.5, along with the number of fragments to handle, and we also compare the same for the attacks runs considering (Knudsen, 2008). We discuss our observations below.

We like to caution and argue that it is not so meaningful to directly compare the final CCR results across two nodes — the characteristics of the technology libraries are quite different in many ways, including the cell types, numbers of metal layers, resistance and capacitance for each layer, design rules, etc., resulting in considerably different physical layouts. For example, we notice that Metal 1 is hardly used and that the timing impact for wires in Metal 1 to Metal 3 is considerable in general for Vashishtha and Clark (2019). Hence most of the regular wires are covered within and above Metal 4, which is a very relevant effect when attacking split manufacturing considering Metal 3 as the split layer, as only a few of the wiring remains disclosed in the FEOL, along with large distances between truly connected fragments. Another related and relevant effect is that the layouts obtained using Vashishtha and Clark (2019) exhibit on average around 5× the number of source pins (and 4× the number of sink pins) when compared to the layouts obtained using Knudsen (2008). Naturally, we can expect that these two effects playing out in conjunction would render any attack on split manufacturing more challenging. Accordingly, we observe that CCR is reduced, namely by 22 percentage points on average. Still, a quite promising result is that for larger designs, such as `b18_C`, our CCR results

obtained for the advanced node even outperform those obtained for the mature node.

6.4.4 Evaluation on Congested Designs

6.4.4.1 Setup

In the fourth set of experiments, we execute our deep learning attack on the Low Density Parity Check (LDPC) benchmark from OpenCores (Montero & Salvadeo, 2013), which is an inherently wire-dominated design and thus suitable for exploration of congestion. We synthesize with a timing constraint of 5 ns (200 MHz) and place and route with a utilization of 15%.

While performing initial experiments, we noticed that the LDPC benchmark was often unroutable, with around 17k DRC violations and formation of many congestion hotspots only after the detailed placement stage. Upon investigation, we could attribute this to large numbers of AOI22 cells which are characterized by very high pin densities, not only inducing congestion but even hindering routing in the vicinity of many instances. Thus, we next employed a setup change as follows: AOI22 cells are disabled during synthesis, but no such restriction is imposed on *Cadence Innovus* during place and route. Doing so restored routability, resulted in DRC-clean layouts, all while allowing for some AOI22 instances to be introduced by layout optimization. Importantly for this set of experiments, the design still remained congested, as confirmed per the congestion maps examined after placement.

We perform iterative synthesis runs to generate ten additional netlist versions, which are functionally equivalent, but exhibit different gate-level implementations. We perform placement and routing for all eleven layouts, and we arrange the layouts into ten for training and cross-validation, and one for testing, respectively.

6.4.4.2 Results

We execute the network-flow attack (Y. Wang et al., 2018) on the LDPC benchmark considering M8 as split layer of M8; this is because the attack runs into time-out considering a split layer of M6. The CCR obtained for the network-flow attack is 28.92%. Recall that, to handle heavily congested layouts, our image-based features are specifically devised to capture routing detours. Thus, our attack achieves a CCR of 39.63%, which is a notable improvement over the network-flow attack.

□ End of chapter.

Chapter 7

Split Manufacturing Defense

Summary

In this chapter, we propose a randomized routing-blockage insertion strategy to escalate the level of layout security against our and other attacks, which can be easily integrated into any commercial physical-design flow. On average, our defense strategy leads to a 22.78 *pp* (percentage points) degradation in CCR when compared with unprotected layouts, while inducing 3.3% and 3.2% overheads on power and timing, respectively, within the same die outlines (zero area cost).

An integrated device manufacturer (IDM) of state-of-the-art requires enormous expenditure in manpower and resources. By 2015, developing a new semiconductor manufacturing facilities costs more than five billion US dollars (Yeh, 2012). Up till 2020, there are only two high-end commercial foundries with volume production on seven nanometer node, which enforces the globalization and diversification of design, synthesis, fabrication, and distribution of integrated circuits (ICs). As

a result, hardware and its implied intellectual properties (IPs) become as vulnerable as software because malicious suppliers are able to steal the entire GDSII layout (Rahman et al., 2020). Attackers may counterfeit defective ICs (G. L. Zhang et al., 2020) or insert hardware Trojans (X. Hu et al., 2020). Split manufacturing delivers the front-end-of-line (FEOL, the transistor layer and a few lower metal layers) to a high-end but untrusted foundry fabricates and integrates the back-end-of-line (BEOL, the remaining upper metal layers) on top of the FEOL by a low-end but trusted facility which is possibly even in-house (Vaidyanathan et al., 2014). In this chapter, we leverage a randomized routing-blockage insertion strategy to defend layout from state-of-the-art attacks. The major contributions of our work are summarized as follows:

- To prevent attackers from learning the behavior of physical design tools, we propose a randomized routing-blockage defense strategy which can be easily integrated into commercial CAD flows.
- We further demonstrate from experiments that the defense strategy is also effective and robust against non-learning-based attacks.

The rest of this chapter is organized as follows. Section 7.1 outlines the threat model and problem formulation. The routing obfuscation strategy is described in Sec. 7.2. The effectiveness of both attack and defense are verified in Sec. 7.3.

7.1 Preliminaries

Consistent with prior work, we assume that attackers have access to the full design information of the FEOL layers including the gates and pins, the related FEOL routing, and the resulting but incomplete netlist formed by source and sink fragments (H. Li, Patnaik, et al., 2019; Y. Wang et al., 2018). Examples of virtual pin pairs are shown by dashed arrows pointing from sink to source fragments in Fig. 6.1. The

objective of our defense is to obfuscate the routing so that the correct connection rate (CCR, Eq. (6.1)) of attacks are minimized.

7.2 Defense against Deep Learning Attack

There are two anti-attack avenues in physical design, namely routing perturbation and placement perturbation. Routing perturbations represent an effective mean for security-aware physical design to protect split-manufactured layouts from proximity attacks (Magaña et al., 2017; Y. Wang et al., 2017). In contrast, placement perturbations can incur large timing overheads or even area overheads, especially for larger designs (Sengupta et al., 2017), and the perturbations are eventually offset in any case, rendering designs vulnerable especially when a higher split layer exposes more information in FEOL (Patnaik, Ashraf, et al., 2018; Patnaik, Knechtel, et al., 2018).

In this work, we seek to defend split-manufactured layouts by randomly inserting routing blockages within the FEOL metal layers. Since commercial tools from leading electronic design automation (EDA) vendors like *Cadence* or *Synopsys* employ deterministic physical-design algorithms, a DNN which is trained on a sufficiently large database of physical layouts can help capturing the essence for the behavior of those tools. Therefore, to ensure that advanced DL-based attacks (or any other attack) cannot easily circumvent the security promises offered by our defense, we seek to introduce "sufficient randomness" for our defense during the layout generation. That is, given the same inputs and constraints, multiple physical-design runs should provide sufficiently different solutions to prevent attackers from learning the defense strategy, yet have to remain fully compliant with design and manufacturing rules which is achieved by virtue of employing commercial-grade tools.

It is understood that randomized routing-level perturbations will have an impact on power, performance, and area (PPA) of the design and, hence, the degree of ran-

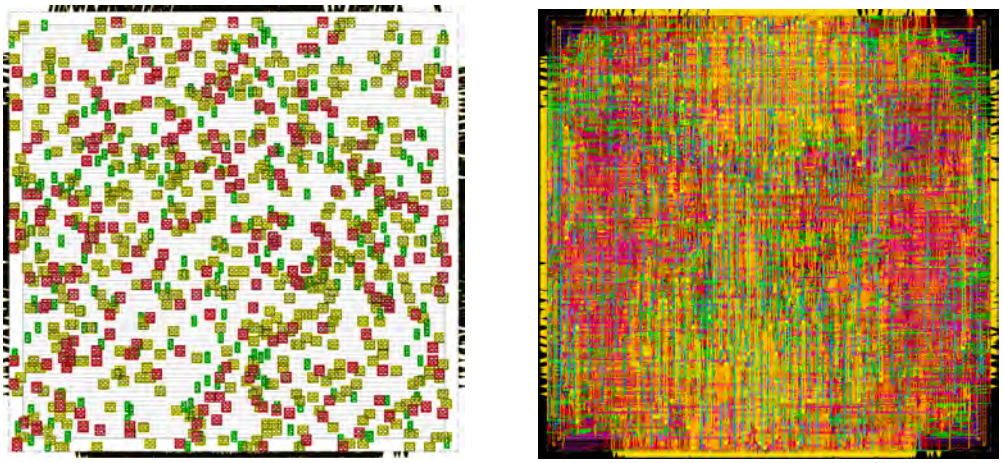


Figure 7.1: Example of routing blockages inserted in ITC-99 benchmark b22_C. (Left) Routing blockages in green are randomly inserted for M3, in yellow for M4, and in red for M5, respectively. Note that colors for the background and for pins at the core boundary are value-inverted for better visibility. (Right) Re-routed layout after blockage insertion.

domness should also remain controllable. Therefore, during the first step of our defense strategy, the designer has to provide the percentage of *g-cells* which shall be blocked at various layers. Assuming a split layer of M6, e.g., a designer should insert blockages throughout any layer(s) of choice below M6. Next, we identify the die and core boundary of the design, and the size of a g-cell. The total number of g-cells is hence derived accordingly for all the layers where the designer seeks to insert blockages. Then, an iterative process is conducted as follows: a random layout location (x, y, z), snapped to the nearest g-cell location, is chosen, and a routing blockage of the same size as the g-cell is introduced into the design. This process is repeated until the blockage requirements specified by the designer are fulfilled. Note that we keep track of the number of blockages already added across the metal layers, also accounting for the preferred routing directions of those layers. We do so to guide the iterative process such that no bias is introduced (by random chance) toward a specific metal layer or a specific routing direction.

Once all routing blockages have been introduced into the design, the global router is invoked again, which then re-routes the blocked parts of all affected nets. Note that we freeze the placement and handle only the routing, to support a fair PPA comparison and a fair security evaluation. Next, we perform design rule check (DRC) for the re-routed solution and, once the design is devoid of any DRC violations, the routing blockages are removed again and the design exchange format (DEF) is generated and streamed out, for attack analysis. In case DRC violations are reported, which is expressed by an overflow of routing resources introduced by some particular blockages, we select among those violating blockages and remove some of them in an iterative manner, until a DRC-clean layout can be obtained. Exemplary layout snapshots for randomly inserted routing blockages and the re-routed, DRC-clean layout for the ITC-99 benchmark `b22_C` are shown in Fig. 7.1.

7.3 Experimental Investigation

We conduct four sets of experiments as follows. In the first set, we illustrate the impact of randomized insertion of routing blockages on attacks (Y. Wang et al., 2018). In the second set, we evaluate the impact of blockages on timing-critical and congested designs. In the third and fourth set of experiments, we analyse the layout cost as induced by routing blockages on regular ITC-99 benchmarks and on timing-critical and congested benchmark versions, respectively.

7.3.1 Routing Perturbation as Defense

7.3.1.1 Setup

In the third set of experiments, we derive six combinational designs from the ITC-99 benchmark suite (Corno et al., 2000). The essence for the layout generation and the deep learning setup is as described in Sec. 6.4.1.1. Furthermore, the procedures for

routing perturbation (Sec. 7.2) are implemented as custom *TCL* scripts working with *Cadence Innovus 17.1*. We consider splitting after M6 and hence insert routing blockages in M3, M4, and M5, respectively. In case a different split layer is chosen by the designer, blockages can be added accordingly. Since we divide the number of blockages across three metal layers, out of which M3 and M5 are horizontal metal layers, and M4 is a vertical layer, more blockages are assigned to M4, while the remaining blockages are distributed uniformly across the horizontal layers (M3 and M5). For the first batch of experiments, labeled as *Fewer Blockages*, we block 12%, 22%, and 12% of the g-cells in M3, M4, and M5, respectively, while in the second batch (*More Blockages*) we block 17%, 25%, and 17% g-cells for the same layers. In general, we add blockages such that timing overheads do not overshoot 5% much, and all layouts are clocked at iso-performance of 5ns. For each design in these two batches, we generate 100 layouts with routing blockages inserted randomly following our proposed defense strategy. As indicated, we ensure that the final layouts after our perturbation procedures remain routable and are devoid of any DRC violations.

We perform a comparative analysis on the randomized insertion of routing blockages leveraging the deep learning attack proposed in Chapter 6 and the network-flow attack (Y. Wang et al., 2018). For our deep learning attack, we consider two different training approaches as follows. In the first approach, we pick forty of the a hundred layouts to train the deep learning model, cross-validate the model using ten other layouts, and attack the remaining fifty unseen layouts; all layouts arising from one design. That is, for each benchmark under attack, a corresponding model is trained individually and that particular model is used only for attacking its respective benchmark. We refer to this as the *robust approach*; it represents the most stringent approach for evaluating the strength of the defense, which can only be conducted by the security-enforcing designers itself, not by an actual attacker. In our second approach, we leverage the leave-one-out scheme which is a standard procedure to sep-

Table 7.1: CCR (%) Comparison with Y. Wang et al. (2018) on Selected Benchmarks

Benchmark	No Blockage		Fewer Blockages			More Blockages		
	TVLSI*	Ours	# Blks	TVLSI*	Ours†	# Blks	TVLSI*	Ours†
b14_C	51.06	80.85	605	55.58	53.61	800	40.10	37.26
b15_C	56.16	60.96	916	31.88	34.50	1216	25.59	29.84
b17_C	24.24	26.07	2377	20.11	25.12	2687	20.90	25.40
b20_C	61.19	72.03	1271	36.12	35.67	1685	29.78	32.00
b21_C	56.40	69.55	1269	41.94	44.53	1682	32.98	34.15
b22_C	47.64	55.36	1874	32.92	34.73	2486	27.36	31.62
Average	49.45	**60.80**	–	36.42	**38.03**	–	29.45	**31.71**

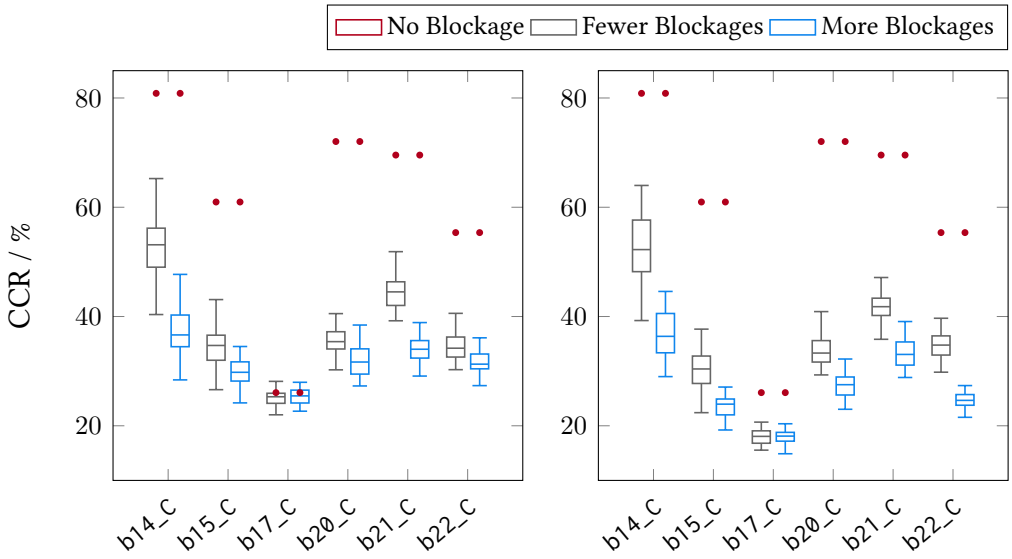
* Y. Wang et al. (2018).
† The CCR for our deep learning attack are obtained using the *robust approach*.

arate training and testing data. Given that we consider six designs in total, we use five designs, with 10 layouts each, to train a model which is then used to attack the one remaining, unseen design. Accordingly, a model is created for each design under attack considering fifty layouts for learning. Note that such an approach can be taken by any attacker.

7.3.1.2 Results

For both the network-flow attack (Y. Wang et al., 2018) and our deep learning attack, we present CCR results for layouts split after M6 in Table 7.1. In presence of the defense, our attack outperforms the network-flow attack in all cases. Next, we describe the findings in more detail.

First, we discuss the security for the *robust* learning approach. Again, we are considering this approach to thoroughly evaluate the strength of our routing-perturbation defense scheme. The corresponding CCR results are illustrated in Table 7.1 and Fig. 7.2 (a). The average CCR results for the layouts with less blocked g-cells ("Fewer Blockages", grey bars) are 53.61%, 34.5%, 25.12%, 35.67%, 44.53%, and 34.73%, respectively. This corresponds to a reduction of CCR by 22.78 percentage points (i.e., the arithmetic dif-



(a) Robust approach. (b) Leave-one-out approach.

Figure 7.2: CCR results for our deep learning attack when splitting after M6, considering selected ITC-99 benchmarks, which are protected with randomly inserted routing blockages. For each benchmark and for each configuration/scenario, we have independently conducted fifty runs; all the results are summarized in boxplots. The upper and lower boundaries of each box span from the 5th to the 95th percentile for the respective data set, while the whiskers represent the minimal and maximal values, the bars inside the boxes represent the median, and the grey dots reflect outliers; all concerning the 50 runs for the respective configuration. Besides, red dots represent the attack results for the respective original, unprotected layouts.

ference of percentage values, abbreviated as *pp*) on average across all benchmarks, when compared to the original, unprotected designs. Once we block even more g-cells ("More Blockages", blue bars), the CCR accuracy drops further: average CCR values are 37.26%, 29.84%, 25.4%, 32%, 34.15%, and 31.62%, respectively. This corresponds to a reduction of CCR by 29.09 *pp* on average across all benchmarks, indicating the strength of the defense even for this theoretical threat model.

Next, we consider the regular *leave-one-out* learning approach where we assume that the design to be attacked is not available for training. The results for both batches of g-cell blockages ("Fewer Blockages" and "More Blockages") are illustrated

in Fig. 7.2 (b). The average CCR results for "Fewer Blockages" (grey bars) are 52.24%, 30.58%, 17.91%, 33.45%, 41.55%, and 34.67%, respectively, for the considered benchmarks. This corresponds to a CCR reduction of 25.74 *pp* on average across all benchmarks. Increasing the number of blockages has a noticeable impact such that the average CCR results for "More Blockages" (blue bars) are 36.13%, 23.92%, 17.68%, 27.81%, 33.52%, and 24.55%, respectively. This corresponds to a CCR reduction of 33.53 *pp* on average across all benchmarks.

Overall, we can see from Fig. 7.2 that the average CCR can be reduced significantly by our randomized routing-perturbation defense. We can see that there is no significant difference for CCR values between the *robust* learning approach and the *leave-one-out* approach, which demonstrates the generality of the learned models across different designs. For larger designs like `b17_C`, however, which are more difficult to attack in general (give the many fragments to be considered), we note that the more blockages under the leave-one-out approach are more challenging to attack. This demonstrates the effectiveness of our defense for large designs under the realistic attack and learning model.

We also perform similar experiments for the network-flow attack (Y. Wang et al., 2018). The corresponding CCR results are illustrated in Fig. 7.3. For the "Fewer Blockages" batch (grey bars), the average CCR values are 55.58%, 31.88%, 20.11%, 36.12%, 41.94%, and 32.92%, respectively. Comparing these with the CCR values observed for original, unprotected layouts, we observe reductions by 13.02 *pp* across all benchmarks. Note that there is no significant reduction for benchmark `b14_C`; this is because the CCR result for the unprotected layout came out lower than expected, to begin with, i.e., at least when considering expectations arising from our deep learning attack (Fig. 7.2). For the "More Blockages" batch (blue bars), as expected, the CCR numbers are further reduced: average CCR results are 40.1%, 25.59%, 20.91%, 29.78%, 32.98%, and 27.36%, respectively. Compared to the original layouts, this setup of blocking more g-cells

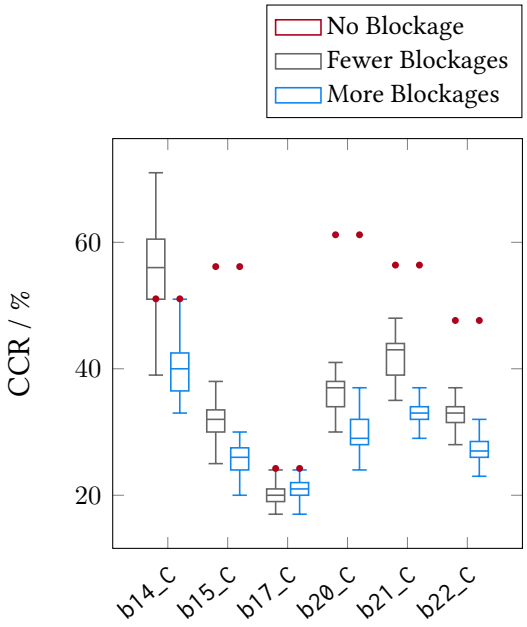


Figure 7.3: CCR results for the network-flow attack (Y. Wang et al., 2018) when splitting after M6, considering selected ITC-99 benchmarks, which are protected with randomly inserted routing blockages. Each scenario for each benchmark considered covers 50 runs. The interpretation of the boxplot is the same as Fig. 7.2.

provides a better protection by enforcing lower CCR by 20.00 *pp* across all benchmarks. When comparing the network-flow attack with our proposed deep learning attack, our method outperforms in all cases, as also shown in Table 7.1.

7.3.2 Defense on Congested Designs

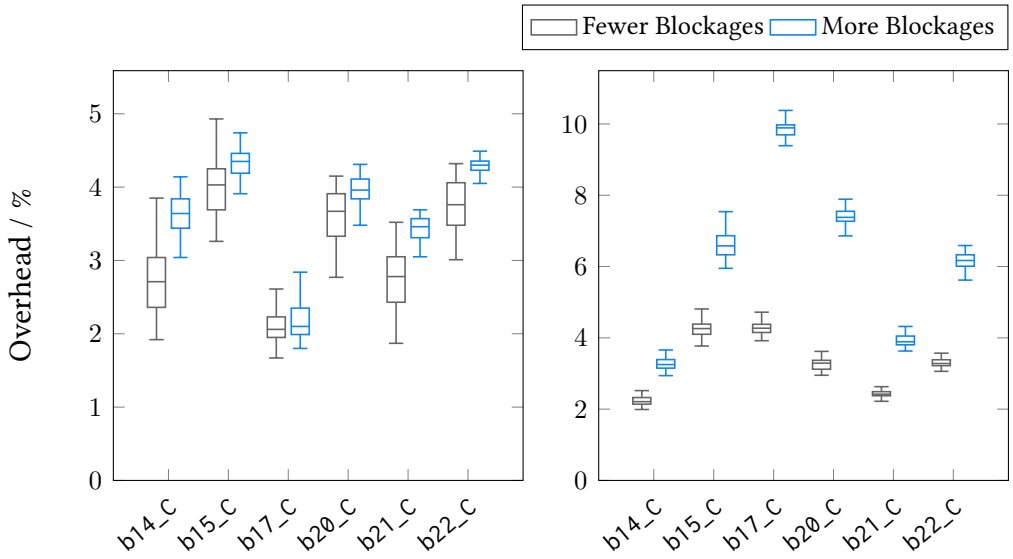
7.3.2.1 Setup

In the fourth set of experiments, we evaluate the impact of the randomized insertion routing blockages on the security of timing-critical and congested designs, respectively. To mimic timing-critical designs, we consider a minor setup revision as follows: we synthesize the selected ITC-99 benchmarks for a 4 ns constraint (250 MHz frequency) instead of the 5 ns (200 MHz) described in Sec. 7.3.1.1. We note that con-

straining at even faster timing (e.g., 3 ns) violated timing paths during synthesis, so 4 ns can be considered as timing-critical setup. With regards to the congested design, we leverage the LDPC design following the same setup as explained in Sec. 6.4.4.1. We consider splitting after M6 for timing-critical ITC-99 benchmarks and insert routing blockages in M3, M4, and M5, respectively, whereas we consider splitting after M8 for the congested design LDPC and accordingly insert blockages in M4, M5, M6, and M7, respectively. We block 12%, 22%, and 12% of the g-cells in M3, M4, and M5, respectively for timing-critical ITC-99 benchmarks, whereas we block 5%, 6%, 6%, and 5% g-cells for M4, M5, M6, and M7, respectively, for the congested LDPC benchmark; these numbers are without loss of generality, but chosen carefully after multiple runs to ensure a DRC-clean layout.

7.3.2.2 Results

The baseline CCR for unprotected layouts is 71.64%, 65.21%, 63.16%, and 56.35% for `b14_C`, `b15_C`, `b20_C`, and `b22_C`, respectively. Upon inserting the randomized routing blockages, we observe an average reduction of 30.37 *pp*, 35.6 *pp*, 25.69 *pp*, and 26.63 *pp*, respectively, in CCR when compared to the baselines. The CCR for the unprotected LDPC benchmark is 28.92%, and invoking our defense strategy helps to reduce the CCR to 25.42%. Although the drop in CCR is only at 3.50 *pp*, our defense helps to increase the absolute number of wrongly inferred connections significantly, and thereby increases the scale of IP protection. For example, splitting the original LDPC benchmark after M8 results in 2,743 cut nets while for our defense technique this increases to 3,629, a difference of 32.3%.



(a) Timing overhead.

(b) Power overhead.

Figure 7.4: Timing and power overheads for selected ITC-99 benchmarks for zero die-area overhead. Each scenario for each benchmark considered covers 50 runs. The interpretation of the boxplots is the same as Fig. 7.2.

7.3.3 Layout Costs Induced by Routing-perturbation Defense

7.3.3.1 Setup

As demonstrated, the proposed routing blockage defense is effective in hindering both the network-flow attack (Y. Wang et al., 2018) and the proposed deep learning attack on split-manufactured designs. Therefore, in this fifth set of experiments, we investigate the timing and power costs incurred by this defense. Recall that we do not incur any overheads for die area. The analysis is carried out for the slow process corner at a supply voltage of 0.95V. To ensure fairness for this layout evaluation (and the above security evaluation), we "freeze" the placement of all the designs and introduce randomized routing blockages to only affect the routing of the layouts. Also, we add blockages such that the timing overheads do not overshoot 5% much and such that no DRC violations occur. All layouts are clocked at iso-performance of 5ns.

7.3.3.2 Results

The timing and power overheads for selected ITC-99 benchmarks are shown in Fig. 7.4. For the "Fewer Blockages" batch, the power overheads are on average 2.24%, 4.21%, 4.29%, 3.27%, 2.43%, and 3.35%, respectively for b14_C, b15_C, b17_C, b20_C, b21_C, and b22_C over original, unprotected layouts. The average timing overheads for the same batch and same set of benchmarks are 2.71%, 3.97%, 2.16%, 3.61%, 2.7%, and 3.76%, respectively. Upon increasing the number of blockages ("More Blockages"), we observe a steady increase in power overheads: the average overheads are now 3.32%, 6.78%, 9.91%, 7.46%, 3.95%, and 6.15%, respectively. This increase is, as expected, particularly pronounced for larger designs like b17_C. The timing overheads, however, increase only marginally, to on average 3.67%, 4.32%, 2.21%, 3.97%, 3.42%, and 4.26%.

Since the insertion of routing blockages forces the router to lift the nets above the split layer and/or detour nets through regions where there are no blockages, an increase in the total count of vias (lifting of nets) and wirelength (detouring of nets) is expected. We confirm this by contrasting the count of total vias and wirelength for the unprotected layouts without blockages and our protected layouts with blockages. Indeed, Section 7.3.3.2 illustrates an increase in vias on selected ITC-99 benchmarks. For the "Fewer Blockages" batch, the increase in total vias are on average 9.37%, 16.38%, 21.7%, 14.28%, 10.82%, and 13.88%, respectively for b14_C, b15_C, b17_C, b20_C, b21_C, and b22_C over original, unprotected layouts. Upon increasing the number of blockages ("More Blockages"), the total increase in total vias for the same set of benchmarks are 14%, 22.42%, 25.52%, 19.79%, 15.78%, and 19.83%, respectively. These numbers attest to the fact that more nets have been lifted above the split layer for both batches/configurations of the proposed routing-perturbation scheme. The increase in individual layer wirelength and total wirelength is shown in Table 7.3. For the "Fewer Blockages" batch, the average increase in total wirelength for the same

Table 7.2: Increase in Via Counts for Our Proposed Defense on Selected ITC-99 Benchmarks

Benchmark				Fewer Blockages			More Blockages		
Name	Total Nets	Placement Util. (%)	Total Vias Before Lifting	Δ_+V67† After Lifting	Δ_+V78† After Lifting	Total Vias After Lifting	Δ_+V67† After Lifting	Δ_+V78† After Lifting	Total Vias After Lifting
b14_C	3009	70	17810	235	50	19478	370	129	20302
b15_C	4306	60	31347	623	313	36481	869	438	38376
b17_C	15477	70	113187	2166	1087	137754	2489	1210	142073
b20_C	7425	70	41455	799	359	47373	1107	556	49658
b21_C	7407	70	41377	684	254	45855	998	447	47907
b22_C	11439	65	62883	1145	497	71614	1603	783	75027

† This denotes the increase in the number of vias when compared to original, unprotected layouts, as averaged over 100 protected layouts.

Table 7.3: Increase in Metal Wirelength for Our Proposed Defense on Selected ITC-99 Benchmarks

Benchmark		Fewer Blockages				More Blockages			
Name	Total Wirelength Before Lifting (μm)	Δ_+M6‡ After Lifting	Δ_+M7‡ After Lifting	Δ_+M8‡ After Lifting	Total Wirelength After Lifting	Δ_+M6‡ After Lifting	Δ_+M7‡ After Lifting	Δ_+M8‡ After Lifting	Total Wirelength After Lifting
b14_C	30540	2309	1537	362	33885	2710	1777	824	35273
b15_C	62470	3780	2676	2031	70193	5256	3415	2930	75012
b17_C	261088	14937	7100	4142	304556	15878	7387	4333	307821
b20_C	79397	5384	5225	2852	94243	7600	6139	4500	101268
b21_C	76678	5336	4418	1810	85760	6734	5340	2769	90437
b22_C	130983	9717	7640	4342	145396	13134	8841	6737	155888

‡ This denotes the increase in the respective metal layer wirelength when compared to original, unprotected layouts, as averaged over 100 protected layouts.

set of benchmarks are 10.95%, 12.36%, 16.65%, 18.69%, 11.84%, and 11%, respectively. As the number of blockages are increased with "More Blockages," the average increase rises to 15.5%, 20.07%, 17.9%, 27.55%, 17.94%, and 19.01%, respectively, which shows that larger parts of the nets reside in the higher layers.

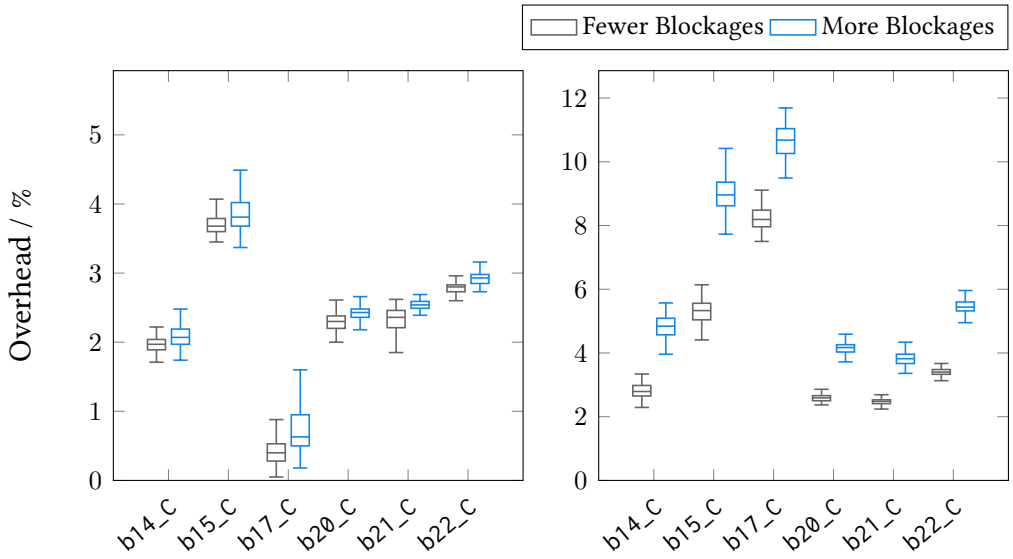
7.3.4 Layout Costs Induced by Defense on Congested Designs

7.3.4.1 Setup

In the sixth set of experiments, we evaluate layout costs for the ITC-99 benchmarks (where timing closure is performed at 4 ns to mimic timing-critical designs) and for the congested LDPC benchmark. For the ITC-99 benchmarks, all designs are generated considering an initial utilization of 70%. For the LDPC benchmark, setup details are the same as in Sec. 6.4.4.1. For both benchmarks, blockages are iteratively added such that timing overheads do not overshoot 5% much and no DRC violations occur.

7.3.4.2 Results

First, we discuss the timing and power overheads for ITC-99 benchmarks considering the aggressive timing closure. As shown in Fig. 7.5(a), the average timing overheads for the same batch and same set of benchmarks are 1.96%, 3.72%, 0.46%, 2.31%, 2.32%, and 2.77%, respectively, for `b14_C`, `b15_C`, `b17_C`, `b20_C`, `b21_C`, and `b22_C` over original, unprotected layouts. Shown in Fig. 7.5(b), the power overheads are on average 2.82%, 5.39%, 8.25%, 2.59%, 2.48%, and 3.42%, respectively. Second, we discuss the overheads incurred for the congested LDPC benchmark. We observe an instance increase of 2.07%, which translates to an increase in standard-cell area of 4.5%; note that this additional standard-cell area does not impact the die area. This increase in instance count and the increase in wirelength (4.49%) leads to a power overhead of 7.1%, albeit at a minimal timing overhead of 0.72%.



(a) Timing overhead.

(b) Power overhead.

Figure 7.5: Overheads for selected ITC-99 benchmarks under aggressive timing constraints (4ns). Each column covers 100 runs of randomized layouts.

Thus, we conclude that our proposed routing-perturbation technique is feasible even for timing-critical and congested designs, and we further conclude that the re-routing required after insertion of routing blockages is naturally imposing power overheads, by virtue of using additional buffer(s) and/or upsizing of standard cells, under the traditional objective of maintaining timing.

□ End of chapter.

Chapter 8

Conclusion

In this thesis, we discuss problems that emerge with the advancement of semiconductor nodes including boosting design rules, heterogeneous standard cells, and globalized manufacturing from the perspective of physical design, attack, and anti-attack.

For detailed placement, we presented a legalization method for mixed-cell-height circuits with consideration of routability constraints like pin access, pin short, edge spacing and fence regions. We proposed a multi-row global legalization that minimizes the total displacement of the cells within a window towards their given global placement positions. We formulated and solved the maximum displacement optimization problem as bipartite matching. In addition, we formulated the fixed row and fixed order optimization problem with a weighted sum of the maximum and average displacement as objective into a min-cost flow problem for further optimization.

We continue our effort for design rule and pin access handling in detailed routing, where a framework with correct-by-construction design rule satisfaction is presented. We demonstrated the handling of pins which have no violation-free on-grid access points. We further illustrated a method of via insertion and via type selection to take the advantage of various vias in the cell library and improve the routing performance. Our proposed detailed router shows superior scalability on all ISPD 2018 and 2019

Initial Detailed Routing Contest designs.

For split manufacturing, we present an attack by deep learning and a defense by routing perturbation. Firstly, we demonstrate vector-based and image-based features and a neural network that can process these heterogeneous features simultaneously. We further propose a SoftMax regression loss that directly enhances the accuracy of the virtual pin pair (VPP) matching problem of split manufacturing and eliminates imbalance issues found in the prior art. Finally, we propose a randomized strategy for routing-blockage insertion that can easily integrated into any commercial physical design flow to protect layouts from being attacked in split manufacturing.

□ End of chapter.

Publication List

Li, H., Chen, G., Jiang, B., Chen, J., & Young, E. F. (2019). Dr. CU 2.0: A scalable detailed routing framework with correct-by-construction design rule satisfaction. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 1–7.

Chen, G., Pui, C.-W., Li, H., & Young, E. F. (2019). Dr. CU: Detailed routing by sparse grid graph and minimum-area-captured path search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*.

Li, H., Patnaik, S., Sengupta, A., Yang, H., Knechtel, J., Yu, B., Young, E. F., & Sinanoglu, O. (2019). Attacking split manufacturing from a deep learning perspective. *ACM/IEEE Design Automation Conference (DAC)*, 1–6.

Chen, G., Pui, C.-W., Li, H., Chen, J., Jiang, B., & Young, E. F. (2019). Detailed routing by sparse grid graph and minimum-area-captured path search. *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 754–760.

Li, H., Chow, W.-K., Chen, G., Young, E. F., & Yu, B. (2018). Routability-driven and fence-aware legalization for mixed-cell-height circuits. *ACM/IEEE Design Automation Conference (DAC)*, 1–6.

Pui, C.-W., Tu, P., Li, H., Chen, G., & Young, E. F. (2018). A two-step search engine for large scale boolean matching under NP3 equivalence. *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 592–598.

Zhang, H., Zhu, F., Li, H., Young, E. F., & Yu, B. (2017). Bilinear lithography hotspot detection. *ACM International Symposium on Physical Design (ISPD)*, 7–14.

Li, H., Hong, W., Liu, Y., & Mou, X. (2017). A novel method of micro-tomography geometric angle calibration with random phantom. *Journal of X-ray Science and Technology*, *25*(4), 641–652.

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, *16*, 265–283. https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

Alpert, C. J., Li, Z., Moffitt, M. D., Nam, G.-J., Roy, J. A., & Tellez, G. (2010). What makes a design difficult to route. *ACM International Symposium on Physical Design (ISPD)*, 7–12. https://doi.org/10.1145/1735023.1735028

Baek, S.-H., Kim, H.-Y., Lee, Y.-K., Jin, D.-Y., Park, S.-C., & Cho, J.-D. (2008). Ultra-high density standard cell library using multi-height cell structure. *Proceedings of SPIE*, *7268*.

Bai, Z.-Z. (2010). Modulus-based matrix splitting iteration methods for linear complementarity problems. *Numerical Linear Algebra with Applications*, *17*(6), 917–933.

Besser, P. (2017). Beol interconnect innovations for improving performance. *NCCAVS Joint Users Group Technical Symposium*.

Bohr, M. T. (1995). Interconnect scaling-the real limiter to high performance ulsi. *IEEE International Electron Devices Meeting (IEDM)*, 241–244.

Brayton, R., & Mishchenko, A. (2010). Abc: An academic industrial-strength verification tool. *International Conference on Computer Aided Verification*, 24–40.

Bustany, I. S., Chinnery, D., Shinnerl, J. R., & Yutsis, V. (2015). ISPD 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement. *ACM International Symposium on Physical Design (ISPD)*, 157–164.

Chang, H.-Y. (2019). Multiple patterning layout decomposition considering complex coloring rules [US Patent 10,395,001].

Chen, G., Pui, C.-W., Chow, W.-K., Lam, K.-C., Kuang, J., Young, E. F., & Yu, B. (2018). RippleFPGA: Routability-driven simultaneous packing and placement for modern FPGAs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, *37*(10), 2022–2035.

Chen, G., Pui, C.-W., Li, H., Chen, J., Jiang, B., & Young, E. F. (2019). Detailed routing by sparse grid graph and minimum-area-captured path search. *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 754–760.

Chen, G., Pui, C.-W., Li, H., & Young, E. F. (2019). Dr. CU: Detailed routing by sparse grid graph and minimum-area-captured path search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*.

Chen, G., & Young, E. F. (2020). SALT: Provably good routing topology by a novel steiner shallow-light tree algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, *39*(6), 1217–1230. https://doi.org/10.1109/TCAD.2019.2894653

Chen, J., Zhu, Z., Zhu, W., & Chang, Y.-W. (2017). Toward optimal legalization for mixed-cell-height circuit designs. *ACM/IEEE Design Automation Conference (DAC)*, 52:1–52:6.

Chen, S., & Vemuri, R. (2018). On the effectiveness of the satisfiability attack on split manufactured circuits. *IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, 83–88.

Chow, W.-K., Pui, C.-W., & Young, E. F. (2016). Legalization algorithm for multiple-row height standard cell design. *ACM/IEEE Design Automation Conference (DAC)*, 83:1–83:6.

Corno, F., Reorda, M. S., & Squillero, G. (2000). RT-level ITC'99 benchmarks and first ATPG results. *IEEE Design & Test of Computers*, *17*(3), 44–53.

Cote, M., Hurat, P., Rieger, M., Miloslavsky, A., & Goinard, D. (2004). How can design for manufacturing improve mask cost and yield? *IEEE/CPMT/SEMI 29th International Electronics Manufacturing Technology Symposium (IEEE Cat. No. 04CH37585)*, 273–276.

Darav, N. K., Bustany, I. S., Kennings, A., & Mamidi, R. (2017). ICCAD-2017 CAD contest in multi-deck standard cell legalization and benchmarks. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 867–871.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, *1*(1), 269–271.

Ding, Y.-X., Chow, W.-K., Posser, G., Yildiz, M. C., & Li, Z. (2020). Circuit design routing based on parallel run length rules [US Patent 10,685,164].

Ding, Y.-X., & Yildiz, M. C. (2020). Circuit design routing using multi-panel track assignment [US Patent 10,706,201].

Dobre, S. A., Kahng, A. B., & Li, J. (2017). Design implementation with noninteger multiple-height cells for improved design quality in advanced nodes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, *37*(4), 855–868.

Dolgov, S., Volkov, A., Wang, L., & Xu, B. (2019). 2019 CAD contest: LEF/DEF based global routing. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 1–4.

Drake, D. E., & Hougardy, S. (2003). A simple approximation algorithm for the weighted matching problem. *Information Processing Letters*, *85*(4), 211–213.

Feng, L., Wang, Y., Hu, J., & Rajendran, J. (2018). *The cat and mouse in split manufacturing*. https://github.com/seth-tamu/network_flow_attack

Flach, G., Fogaça, M., Monteiro, J., Johann, M., & Reis, R. (2017). Rsyn: An extensible physical synthesis framework. *ACM International Symposium on Physical Design (ISPD)*, 33–40.

Frederick, M. (2014). Poly pitch and standard cell co-optimization below 28nm. *IEEE International Electron Devices Meeting (IEDM)*.

Gessler, F., Brisk, P., & Stojilovic, M. (2020). A shared-memory parallel implementation of the RePlAce global cell placer. *International Conference on VLSI Design*, (CONF).

Gester, M., Müller, D., Nieberg, T., Panten, C., Schulte, C., & Vygen, J. (2013). BonnRoute: Algorithms and data structures for fast and good VLSI routing. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, *18*(2), 1–24.

Gonçalves, S. M., da Rosa, L. S., & Marques, F. d. S. (2019). An improved heuristic function for A*-based path search in detailed routing. *IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5.

Griffiths, D., & Boehm, J. (2019). A review on deep learning techniques for 3D sensed data classification. *Remote Sensing*, *11*(12), 1499.

Hansen, M. C., Yalcin, H., & Hayes, J. P. (1999). Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering. *IEEE Design & Test of Computers*, *16*(3), 72–80.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1972). Correction to "a formal basis for the heuristic determination of minimum cost paths". *ACM SIGART Bulletin*, (37), 28–29.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Heo, S. i., Kahng, A. B., Kim, M., Wang, L., & Yang, C. (2019). Detailed placement for ir drop mitigation by power staple insertion in sub-10nm vlsi. *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 830–835.

Hetzel, A. (1998). A sequential detailed router for huge grid graphs. *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 332–338.

Hill, B., Karmazin, R., Otero, C. T. O., Tse, J., & Manohar, R. (2013). A split-foundry asynchronous FPGA. *IEEE Custom Integrated Circuits Conference (CICC)*, 1–4.

Hill, D. (2002). Method and system for high speed detailed placement of cells within an integrated circuit design [US Patent 6,370,673].

Hopcroft, J. E., & Karp, R. M. (1973). An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing (SICOMP)*, *2*(4), 225–231.

Hu, S., & Hu, J. (2007). Pattern sensitive placement for manufacturability. *ACM International Symposium on Physical Design (ISPD)*, 27–34.

Hu, X., Zhao, Y., Deng, L., Liang, L., Zuo, P., Ye, J., Lin, Y., & Xie, Y. (2020). Practical attacks on deep neural networks by memory Trojaning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 1–1. https://doi.org/10.1109/TCAD.2020.2995347

Hung, C.-Y., Chou, P.-Y., & Mak, W.-K. (2017). Mixed-cell-height standard cell placement legalization. *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, 149–154.

Jhaveri, T., Rovner, V., Liebmann, L., Pileggi, L., Strojwas, A. J., & Hibbeler, J. D. (2010). Co-optimization of circuits, layout and lithography for predictive technology scaling beyond gratings. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, *29*(4), 509–527.

Kahng, A. B., Lienig, J., Markov, I. L., & Hu, J. (2011). *VLSI physical design: From graph partitioning to timing closure*. Springer Science & Business Media.

Kahng, A. B., Wang, L., & Xu, B. (2020a). The Tao of PAO: Anatomy of a pin access oracle for detailed routing. *ACM/IEEE Design Automation Conference (DAC)*, 1–6.

Kahng, A. B., Wang, L., & Xu, B. (2020b). TritonRoute: The open source detailed router. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*. https://doi.org/10.1109/TCAD.2020.3003234

Khasawneh, M., & Madden, P. H. (2020). Hill climbing with trees: Detail placement for large windows. *ACM International Symposium on Physical Design (ISPD)*, 9–16.

Király, Z., & Kovács, P. (2012). Efficient implementations of minimum-cost flow algorithms. *arXiv preprint arXiv:1207.6381*.

Knudsen, J. (2008). NanGate 45nm open cell library. *CDNLive, EMEA*.

Laws, D. A. (2010). A company of legend: The legacy of fairchild semiconductor. *IEEE Annals of the History of Computing*, *32*(1), 60–74.

Lee, M. C., Jan, G. E., & Luo, C. C. (2020). An efficient rectilinear and octilinear steiner minimal tree algorithm for multidimensional environments. *IEEE Access*, *8*, 48141–48150.

Li, H., Chen, G., Jiang, B., Chen, J., & Young, E. F. (2019). Dr. CU 2.0: A scalable detailed routing framework with correct-by-construction design rule satisfaction. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 1–7.

Li, H., Chow, W.-K., Chen, G., Young, E. F., & Yu, B. (2018). Routability-driven and fence-aware legalization for mixed-cell-height circuits. *ACM/IEEE Design Automation Conference (DAC)*, 1–6.

Li, H., Hong, W., Liu, Y., & Mou, X. (2017). A novel method of micro-tomography geometric angle calibration with random phantom. *Journal of X-ray Science and Technology*, *25*(4), 641–652.

Li, H., Patnaik, S., Sengupta, A., Yang, H., Knechtel, J., Yu, B., Young, E. F., & Sinanoglu, O. (2019). Attacking split manufacturing from a deep learning perspective. *ACM/IEEE Design Automation Conference (DAC)*, 1–6.

Li, M., Yu, B., Lin, Y., Xu, X., Li, W., & Pan, D. Z. (2018). A practical split manufacturing framework for trojan prevention via simultaneous wire lifting and cell insertion. *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 265–270.

Li, W., Xia, J., Ma, Y., Li, J., Lin, Y., & Yu, B. (2020). Adaptive layout decomposition with graph embedding neural networks. *ACM/IEEE Design Automation Conference (DAC)*.

Li, X., Chen, J., Zhu, W., & Chang, Y.-W. (2019). Analytical mixed-cell-height legalization considering average and maximum movement minimization. *ACM International Symposium on Physical Design (ISPD)*, 27–34.

Lin, Y.-H., Yu, B., Z. Pan, D., & Li, Y.-L. (2012). TRIAD: A triple patterning lithography aware detailed router. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 123–129.

Lin, Y., Yu, B., Xu, X., Gao, J.-R., Viswanathan, N., Liu, W.-H., Li, Z., Alpert, C. J., & Pan, D. Z. (2018). MrDP: Multiple-row detailed placement of heterogeneous-sized cells for advanced nodes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, *37*(6), 1237–1250.

Liu, J., Pui, C.-W., Wang, F., & Young, E. F. (2020). CUGR: Detailed-routability-driven 3d global routing with probabilistic resource model. *ACM/IEEE Design Automation Conference (DAC)*, 1–6.

Liu, W.-H., Mantik, S., Chow, W.-K., Ding, Y., Farshidi, A., & Posser, G. (2019). ISPD 2019 initial detailed routing contest and benchmark with advanced routing rules. *ACM International Symposium on Physical Design (ISPD)*, 147–151.

Ma, X., Zheng, X., & Arce, G. R. (2020). Fast inverse lithography based on dual-channel model-driven deep learning. *Optics Express*, *28*(14), 20404–20421.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. *International Conference on Machine Learning (ICML)*, *30*, 3.

Magaña, J., Shi, D., Melchert, J., & Davoodi, A. (2017). Are proximity attacks a threat to the security of split manufacturing of integrated circuits? *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, *25*(12), 3406–3419.

Mantik, S., Posser, G., Chow, W.-K., Ding, Y., & Liu, W.-H. (2018). ISPD 2018 initial detailed routing contest and benchmarks. *ACM International Symposium on Physical Design (ISPD)*, 140–143.

Maßberg, J., & Nieberg, T. (2013). Rectilinear paths with minimum segment lengths. *Discrete Applied Mathematics*, *161*(12), 1769–1775.

McCants, C. (2016). *Trusted integrated chips (TIC) program*. https://www.ndia.org/-/media/sites/ndia/meetings-and-events/divisions/systems-engineering/past-events/trusted-micro/2016-august/mccants-carl.ashx

Montero, M. A., & Salvadeo, P. A. (2013). Opencores® a robust complex network. *2013 7th Argentine School of Micro-Nanoelectronics, Technology and Applications*, 15–21.

Moore, G. E. et al. (1975). Progress in digital integrated electronics. *IEEE International Electron Devices Meeting (IEDM)*, *21*, 11–13.

Nieberg, T. (2011). Gridless pin access in detailed routing. *ACM/IEEE Design Automation Conference (DAC)*, 170–175.

O'Brien, F. (2010). *The apollo guidance computer: Architecture and operation*. Springer Science & Business Media.

OECD, & EUIPO. (2019). *Trends in trade in counterfeit and pirated goods*. OECD Publishing. https://doi.org/10.1787/g2g9f533-en

Otto, O. W., Garofalo, J. G., Low, K. K., et al. (1994). Automated optical proximity correction: A rules-based approach. *Proceedings of SPIE*, 278–293.

Ozdal, M. M. (2009). Detailed-routing algorithms for dense pin clusters in integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, *28*(3), 340–349.

Pan, D. Z., Cho, M., & Yuan, K. (2010). *Manufacturability aware routing in nanometer vlsi*. Now Publishers Inc.

Park, D., Kang, I., Kim, Y., Gao, S., Lin, B., & Cheng, C.-K. (2019). ROAD: Routability analysis and diagnosis framework based on sat techniques. *ACM International Symposium on Physical Design (ISPD)*.

Park, D., Lee, D., Kang, I., Gao, S., Lin, B., & Cheng, C.-K. (2020). Sp&r: Simultaneous placement and routing framework for standard cell synthesis in sub-7nm. *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 345–350.

Patnaik, S., Ashraf, M., Knechtel, J., & Sinanoglu, O. (2018). Raise your game for split manufacturing: Restoring the true functionality through BEOL. *ACM/IEEE Design Automation Conference (DAC)*, 140:1–140:6. https://doi.org/10.1145/3195970.3196100

Patnaik, S., Knechtel, J., Ashraf, M., & Sinanoglu, O. (2018). Concerted wire lifting: Enabling secure and cost-effective split manufacturing. *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 251–258.

Perez, T. D., & Pagliarini, S. (2020). A survey on split manufacturing: Attacks, defenses, and challenges. *arXiv preprint*.

Peyer, S., Rautenbach, D., & Vygen, J. (2009). A generalization of dijkstra's shortest path algorithm with applications to vlsi routing. *Journal of Discrete Algorithms*, *7*(4), 377–390.

Polanyi, M. (2009). *The tacit dimension*. University of Chicago press.

Pui, C.-W., Tu, P., Li, H., Chen, G., & Young, E. F. (2018). A two-step search engine for large scale boolean matching under NP3 equivalence. *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 592–598.

Qi, Z.-D., Cai, Y.-C., & Zhou, Q. (2015). Design-rule-aware congestion model with explicit modeling of vias and local pin access paths. *Journal of Computer Science and Technology*, *30*(3), 614–628.

Qiu, X., & Marek-Sadowska, M. (2012). Can pin access limit the footprint scaling? *ACM/IEEE Design Automation Conference (DAC)*, 1100–1106.

Rahman, M. T., Rahman, M. S., Wang, H., Tajik, S., Khalil, W., Farahmandi, F., Forte, D., Asadizanjani, N., & Tehranipoor, M. (2020). Defense-in-depth: A recipe for logic locking to prevail. *Integration, the VLSI Journal*.

Rajendran, J., Sinanoglu, O., & Karri, R. (2013). Is split manufacturing secure? *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 1259–1264.

Samanta, T., Rahaman, H., & Dasgupta, P. (2011). Near-optimal y-routed delay trees in nanometric interconnect design. *IET Computers & Digital Techniques*, *5*(1), 36–48.

Santos, C., Reis, R., Godoi, G., Barros, M., & Duarte, F. (2012). Multi-bit flip-flop usage impact on physical synthesis. *2012 25th Symposium on Integrated Circuits and Systems Design (SBCCI)*, 1–6.

Schellenberg, F. (2003). A little light magic [optical lithography]. *IEEE Spectrum*, *40*(9), 34–39.

Sechen, C., & Sangiovanni-Vincentelli, A. (1986). Timberwolf3. 2: A new standard cell placement and global routing package. *ACM/IEEE Design Automation Conference (DAC)*, 432–439.

Sengupta, A., Patnaik, S., Knechtel, J., Ashraf, M., Garg, S., & Sinanoglu, O. (2017). Rethinking split manufacturing: An information-theoretic approach with secure layout techniques. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 329–326.

Sherazi, S., Chae, J., Debacker, P., Matti, L., Verkest, D., Mocuta, A., Kim, R., Spessot, A., Dounde, A., & Ryckaert, J. (2019). CFET standard-cell design down to 3Track height for node 3nm and below. *Proceedings of SPIE*, *10962*, 16–27. https://doi.org/10.1117/12.2514571

Sherlekar, D. D. (2014). Cell architecture for increasing transistor size [US Patent 8,631,374].

Shin, H., & Sangiovanni-Vincentelli, A. (1987). A detailed router based on incremental routing modifications: Mighty. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, *6*(6), 942–955.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, *362*(6419), 1140–1144.

Spindler, P., Schlichtmann, U., & Johannes, F. M. (2008). Abacus: Fast legalization of standard cell circuits with minimal movement. *ACM International Symposium on Physical Design (ISPD)*, 47–53.

Sun, F.-K., Chen, H., Chen, C.-Y., Hsu, C.-H., & Chang, Y.-W. (2018). A multithreaded initial detailed routing algorithm considering global routing guides. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.

Sun, G., Huang, P., Li, J., & Arifin, T. (2016). Methods, systems, and articles of manufacture for enhancing metrics of electronic designs using design rule driven physical design implementation techniques [US Patent 9,372,952].

Vaidyanathan, K., Das, B. P., Sumbul, E., Liu, R., & Pileggi, L. (2014). Building trusted ICs using split fabrication. *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, 1–6.

Vashishtha, V., & Clark, L. T. (2019). A finfet-based framework for vlsi design at the 7 nm node. *Energy Efficient Computing & Electronics: Devices to Systems*, 1.

Vygen, J. (1998). Algorithms for detailed placement of standard cells. *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 321–324.

Wang, C.-H., Wu, Y.-Y., Chen, J., Chang, Y.-W., Kuo, S.-Y., Zhu, W., & Fan, G. (2017). An effective legalization algorithm for mixed-cell-height standard cells. *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 450–455.

Wang, Y., Chen, P., Hu, J., Li, G., & Rajendran, J. (2018). The cat and mouse in split manufacturing. *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, *26*(5), 805–817.

Wang, Y., Chen, P., Hu, J., & Rajendran, J. ((2017). Routing perturbation for enhanced security in split manufacturing. *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 605–510.

Wong, A. K. (2003). Microlithography: Trends, challenges, solutions, and their impact on design. *IEEE Micro*, *23*(2), 12–21.

Wu, G., & Chu, C. (2016). Detailed placement algorithm for VLSI design with double-row height standard cells. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, *35*(9), 1569–1573.

Xu, X., Yu, B., Gao, J.-R., Hsu, C.-L., & Pan, D. Z. (2016). PARR: Pin-access planning and regular routing for self-aligned double patterning. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, *21*(3), 42:1–42:21.

Yang, S. (1991). *Logic synthesis and optimization benchmarks user guide: Version 3.0.* Microelectronics Center of North Carolina (MCNC).

Yeh, A. (2012). Trends in the global IC design service market. *DIGITIMES research.* http://www.digitimes.com/news/a20120313RS400

Yoshimura, T., & Kuh, E. S. (1982). Efficient algorithms for channel routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, *1*(1), 25–35.

Yu, B., Yuan, K., Ding, D., & Z. Pan, D. (2015). Layout decomposition for triple patterning lithography. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, *34*(3), 433–446.

Yu, B., Chang, L., Ahmed, S., Wang, H., Bell, S., Yang, C.-Y., Tabery, C., Ho, C., Xiang, Q., King, T.-J., Bokor, J., Hu, C., Lin, M.-R., & Kyser, D. (2002). FinFET scaling to 10 nm gate length. *IEEE International Electron Devices Meeting (IEDM)*, 251–254. https://doi.org/10.1109/IEDM.2002.1175825

Yutsis, V., Bustany, I. S., Chinnery, D., Shinnerl, J. R., & Liu, W.-H. (2014). ISPD 2014 benchmarks with sub-45nm technology rules for detailed-routing-driven placement. *ACM International Symposium on Physical Design (ISPD)*, 161–168.

Zeng, W., Zhang, B., & Davoodi, A. (2019). Analysis of security of split manufacturing using machine learning. *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, *27*(12), 2767–2780.

Zhang, G. L., Li, B., Li, M., Yu, B., Pan, D. Z., Brunner, M., Sigl, G., & Schlichtmann, U. (2020). TimingCamouflage+: Netlist security enhancement with unconventional timing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 1–1.

Zhang, H., Zhu, F., Li, H., Young, E. F., & Yu, B. (2017). Bilinear lithography hotspot detection. *ACM International Symposium on Physical Design (ISPD)*, 7–14.

Zhang, X., Zhuang, Z., Liu, G., Huang, X., Liu, W.-H., Guo, W., & Wang, T.-C. (2020). Minidelay: Multi-strategy timing-aware layer assignment for advanced technology nodes. *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 586–591.

Zhang, Y., & Chu, C. (2013). RegularRoute: An efficient detailed router applying regular routing patterns. *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, *21*(9), 1655–1668.

Zhou, Z., Zhu, Z., Chen, J., Ma, Y., Yu, B., Ho, T.-Y., Lemieux, G., & Ivano, A. (2019). Congestion-aware global routing using deep convolutional generative adversarial networks. *ACM/IEEE Workshop on Machine Learning CAD (MLCAD)*.

Zhu, Z., Chen, J., Zhu, W., & Chang, Y.-W. (2020). Mixed-cell-height legalization considering technology and region constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*.

Zhu, Z., Huang, Z., Yang, P., Zhu, W., Chen, J., Zhou, H., & Dong, S. (2020). Mixed-cell-height legalization considering complex minimum width constraints and half-row fragmentation effect. *Integration, the VLSI Journal*, *71*, 1–10.