UNIVERSITY OF CALIFORNIA, SAN DIEGO

Floorplan Representation, Global Placement, and Routability Analysis for VLSI Layout Design Automation

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science (Computer Engineering)

by

Ilgweon Kang

Committee in charge:

Professor Chung-Kuan Cheng, Chair
Professor Ronald Graham
Professor Andrew B. Kahng
Professor Farinaz Koushanfar
Professor Bill Lin

2018

The dissertation of Ilgweon Kang is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2018

DEDICATION

I dedicate this thesis to my family.
Without their support this thesis would not have been finished.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Foremost, I would like to thank my advisor Professor Chung-Kuan Cheng for his encouragement and support. Without his generous advice my Ph.D. study would not have been finished. Indeed, I have learned a lot from him not only his brilliant academic knowledge, also his attitude toward research and life.

I would like to thank to my father Kyungchan Kang and my mother Woesoon Kim for their devotion and support. I would like to thank to my father-in-law Jongkyun Noh and my mother-in-law Kyoungock Jin for their encouragement. And I would like to thank to my wife Heemin Noh for her selfless support and my son Ryan Hyunwoo Kang for his assistance. My research would not have been possible without their sacrifice.

I would like to thank Professor Andrew B. Kahng for his academical support and encouragement. I have been truly influenced by his passionate and responsible attitude toward research.

I would like to thank my labmates (Xinyuan Wang, Po-Ya Hsu, Dongwon Park), former labmates (Dr. Hao Zhuang, Dr. Jingwei Lu, Professor Seokhyeong Kang, Dr. Tuck-Boon Chan, Dr. Siddhartha Nath, Dr. Jiajia Li, Dr. Wei-Ting (Jonas) Chan, Hyein Lee, Kwangsoo Han, Jeng-Hau Lin, Xinan Wang) and schoolmates (Lutong Wang, Mulong Luo, Bangqi Xu, Yeseong Kim, Xun Jiao, Dr. Seungjin Na) for their support and encouragement.

I would like to thank my industrial collaborators (Nancy MacDonald, Changho Han) for their insightful feedback and comments in my researches. I would like to thank my colleagues (Yufeng Luo, Dr. Chin-Chih Chang, Dr. Min Ouyang) for their thoughtful support and invaluable guidance.

I sincerely appreciate my thesis committee members Professor Ronald Graham, Professor Farinaz Koushanfar and Professor Bill Lin for their time and insightful feedback and comments.

The material in this thesis is based on the following publications.

Chapter 1 contains a reprint of Ilgweon Kang and Chung-Kuan Cheng, “Physical Layout after Half a Century: From Back-Board Ordering to Multi-Dimensional Placement and Beyond”, *Proc. ACM/IEEE International Symposium on Physical Design*, 2017. The dissertation author is the primary author of the paper.

Chapter 2 contains a reprint of Fang Qiao, Ilgweon Kang, Daniel Kane, Evangeline Fung Yu Young, Chung-Kuan Cheng and Ronald Graham, "3D Floorplan Representations: Corner Links and Partial Order", *Proc. IEEE International Conference of 3D System Integration*, 2016. Chapter 2 also contains the draft submitted to *ACM Transactions on Design Automation of Electronic Systems*, Ilgweon Kang, Fang Qiao, Dongwon Park, Daniel Kane, Evangeline Fung Yu Young, Chung-Kuan Cheng and Ronald Graham, "3-D Floorplan Representations by Using Corner Links and Partial Order". The dissertation author is a main contributor of the paper and the primary author of the submitted draft.

Chapter 3 contains the draft submitted to *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Chung-Kuan Cheng, Andrew B. Kahng, Ilgweon Kang and Lutong Wang, "Advancing Solution Quality and Routability Validation in Global Placement", 2017. The dissertation author is the primary author of the submitted draft.

Chapter 4 contains a reprint of Ilgweon Kang, Dongwon Park, Changho Han and Chung-Kuan Cheng, "Fast and Precise Routability Analysis with Conditional Design Rules", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2018. The dissertation author is the primary author of the paper.

My coauthors (Professor Chung-Kuan Cheng, Professor Ronald Graham, Changho Han, Professor Andrew B. Kahng, Professor Daniel Kane, Dongwon Park, Fang Qiao, Lutong Wang, and Professor Evangeline F. Y. Young, listed in alphabetical order) have all kindly approved the inclusion of the aforementioned publications in my thesis.

VITA

2006	B.Sc., Electrical and Electronics Engineering, Yonsei University, Seoul, Republic of Korea
2008	M.Sc., Electrical and Electronics Engineering, Yonsei University, Seoul, Republic of Korea
2008-2010	Engineer, DRAM Development Division, SK Hynix, Icheon, Republic of Korea
2010-2012	Engineer, R&D Division, SK Hynix, Icheon, Republic of Korea
2016	C.Phil., Computer Science (Computer Engineering), University of California, San Diego
2017-Present	Lead Software Engineer, Digital Signoff Group, Cadence Design Systems Inc., San Jose
2018	Ph.D., Computer Science (Computer Engineering), University of California, San Diego

All papers co-authored with Professor Andrew B. Kahng and the paper published at International Symposium on Physical Design (2018) have authors listed in alphabetical order. The first two drafts in the list are submitted for publication. The last three papers are published when the dissertation author was pursuing his Master's degree in Yonsei university, Seoul, Republic of Korea.

- **Ilgweon Kang**, Fang Qiao, Dongwon Park, Daniel Kane, Evangeline Fung Yu Young, Chung-Kuan Cheng and Ronald Graham, "3-D Floorplan Representations by Using Corner Links and Partial Order", *ACM Transactions on Design Automation of Electronic Systems*, under review.

- Chung-Kuan Cheng, Andrew B. Kahng, **Ilgweon Kang** and Lutong Wang, "Advancing Solution Quality and Routability Validation in Global Placement", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, under review.

- **Ilgweon Kang**, Dongwon Park, Changho Han and Chung-Kuan Cheng, "Fast and Precise Routability Analysis with Conditional Design Rules", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2018, to appear.

- Chung-Kuan Cheng, Ronald Graham, **Ilgweon Kang**, Dongwon Park, Xinyuan Wang, "Tree Structures and Algorithms for Physical Design", *Proc. ACM/IEEE International Symposium on Physical Design*, 2018, pp. 120-125.

- **Ilgweon Kang** and Chung-Kuan Cheng, "Physical Layout after Half a Century: From Back-Board Ordering to Multi-Dimensional Placement and Beyond", *Proc. ACM/IEEE International Symposium on Physical Design*, 2017, pp. 123-128.

- Fang Qiao, **Ilgweon Kang**, Daniel Kane, Chung-Kuan Cheng and Ronald Graham, "3D Floorplan Representations: Corner Links and Partial Order", *Proc. IEEE International Conference of 3D System Integration*, 2016, pp. 1-5.

- Hao Zhuang, Wenjian Yu, Shih-Hung Weng, **Ilgweon Kang**, Jeng-Hau Lin, Xiang Zhang, Ryan Coutts and Chung-Kuan Cheng, "Simulation Algorithms with Exponential Integration for Time-Domain Analysis of Large-Scale Power Delivery Networks", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35(10) (2016), pp. 1681-1694.

- Jingwei Lu, Hao Zhuang, **Ilgweon Kang**, Pengwen Chen and Chung-Kuan Cheng, "ePlace-3D: Electrostatics based Placement for 3D-ICs", *Proc. ACM/IEEE International Symposium on Physical Design*, 2016, pp. 11-18.

- Jeng-Hau Lin, Hao Liu, Chia-Hung Liu, Phillip Lam, Gung-Yu Pan, Hao Zhuang, **Ilgweon Kang**, Patrick P. Mercier and Chung-Kuan Cheng, "An Interdigitated Non-Contact ECG Electrode for Impedance Compensation and Signal Restoration", *Proc. IEEE Biomedical Circuits and Systems Conference*, 2015, pp. 543-546.

- Hao Zhuang, Wenjian Yu, **Ilgweon Kang**, Xinan Wang and Chung-Kuan Cheng, "An Algorithmic Framework of Large-Scale Circuit Simulation Using Exponential Integrators", *Proc. ACM/IEEE Design Automation Conference*, 2015, pp. 163:1-163:6.

- Hao Zhuang, Xinan Wang, **Ilgweon Kang**, Jeng-Hau Lin and Chung-Kuan Cheng, "Dynamic Analysis of Power Delivery Network with Nonlinear Components Using Matrix Exponential Method", *Proc. IEEE Symposium on Electromagnetic Compatibility and Signal Integrity*, 2015, pp. 248-252.

- **Ilgweon Kang**, Xinan Wang, Jeng-Hau Lin, Ryan Coutts and Chung-Kuan Cheng, "Impulse Response Generation from S-Parameters for Power Delivery Network Simulation", *Proc. IEEE Symposium on Electromagnetic Compatibility and Signal Integrity*, 2015, pp. 277-282.

- Tuck-Boon Chan, Andrew B. Kahng, **Ilgweon Kang**, Hyein Lee and Jiajia Li, "Toward Assessment of True 3D Design Benefits over 2D Implementation", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2014, poster presentation.

- Andrew B. Kahng and **Ilgweon Kang**, "Co-Optimization of Memory BIST Grouping, Test Scheduling, and Logic Placement", *Proc. ACM/IEEE Design, Automation and Test in Europe*, 2014, pp. 196:1-196:6.

- Andrew B. Kahng, **Ilgweon Kang** and Siddhartha Nath, "Incremental Multiple-Scan Chain Ordering for ECO Flip-Flop Insertion", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2013, pp. 705-712.

- Woosik Jeong, **Ilgweon Kang**, Kyowon Jin and Sungho Kang, "A Fast Bulit-in Redundancy Analysis for Memories with Optimal Repair Rate Using a Line-based Search Tree", *IEEE Transactions on Very Large Scale Integration Systems* 17(12) (2009), pp. 1665-1678.

- **Ilgweon Kang**, Woosik Jeong and Sungho Kang, "High-Efficiency Memory BISR with Two Serial RA Stages Using Spare Memories", *IET Electronics Letters* 44(8) (2008), pp. 515-517.

- **Ilgweon Kang** and Sungho Kang, "The Efficient Memory BISR Architecture Using Sign Bits", *Journal of the Institute of Electronics Engineers of Korea SD* 44(12) (2007), pp. 85-92.

ABSTRACT OF THE DISSERTATION

Floorplan Representation, Global Placement, and Routability Analysis for VLSI Layout Design Automation

by

Ilgweon Kang

Doctor of Philosophy in Computer Science (Computer Engineering)

University of California, San Diego, 2018

Professor Chung-Kuan Cheng, Chair

In the past decades, semiconductor technologies have significantly contributed to the modern society and human welfare, and led entire industries toward more automated systems with advancement of integrated circuits (ICs). Innovations and achievements on physical design (PD) have guided progresses of modern VLSI designs and automation. With the advanced performance of ICs, the overall industry has been profitable, and the global semiconductor market size has shown an upward-climbing trend for the past decades. Consequently, high-performance ICs enable the recent advent of the fourth industrial revolution that evolves almost every industry, e.g., artificial intelligence (AI), the Internet of things (IoT), bio- and nano-technology, autonomous vehicles, robotics, etc.

While "Moore's Law" and "Dennard Scaling" have shown the correction of the slowing down, designing IC has become much more sophisticated and complicated. IC layout design directly impacts on timing closure, die utilization, routability, and design turnaround time (TAT); these in turn affect the classic design metrics of operating frequency, yield, power consumption and cost. As a result, physical design engineers face many nontrivial challenges, and the overall design cost increases rapidly. The industries look for higher efficiency in design optimization, automation, and innovation for high-performance ICs and design-cost reduction.

This dissertation describes new design methodologies for the advanced IC layout development and design automation. Chapter 2 introduces new three-dimensional (3-D) floorplan representations, *corner links*, *four trees*, and *partial order*, which enhance 3-D IC physical design automation. Our floorplan representations are potentially extendable toward multiple dimensions by adding factors such as time, energy, temperature, security, etc. Chapter 3 describes our constraint-driven and routability-driven global placement engine, *RePlAce*. RePlAce is a flat, nonlinear analytical global placement engine with electrostatics-based global-smooth density cost function. RePlAce addresses routing congestion as well as classical design goals (such as wirelength, area, etc.) with analogy of charge and electrical potential distribution. Chapter 4 presents a new framework that quickly identifies the design rule-correct routability through well-organized Boolean satisfiability (SAT) formulation. Our routability analysis method is developed on top of the multi-commodity flow theory and SAT-friendly ILP-based (integer linear programming-based) detailed routing formulation.

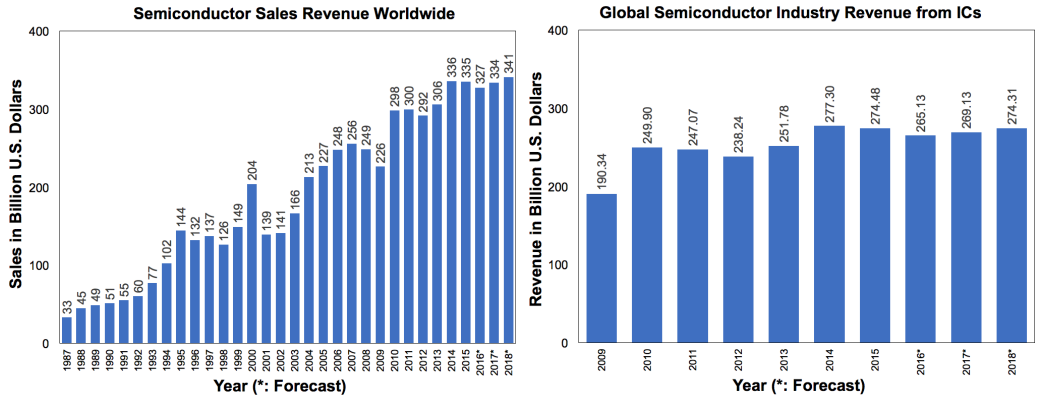
Chapter 1

Introduction

In the past decades, semiconductor technologies have significantly contributed to the modern society and human welfare, and led entire industries toward more automated systems with advancement of integrated circuits (ICs). Innovations and advancements on physical design (PD) guide progresses of modern VLSI designs and the automation and deserve considerable credits on the reduction of design gaps of VLSI layout designs. By virtue of the PD community and their efforts, the performance of ICs has been extremely advanced, the overall industry has been profitable, and the global semiconductor market size has shown an upward climbing for the past decades (as shown in Figures 1.1(a)-(b)).

While "Moore's Law" and "Dennard Scaling" have shown the correction of the slowing down, the hardware design cost increases rapidly and the cost per gate trend reverses from downward decreasing toward upward increasing after 20*nm* technology node. Figure 1.2 shows the hardware design cost per technology node in bar chart and the cost per gate trend in a curved line. In other words, the cost reduction by scaling becomes harder and harder or no longer available. In the meantime, the growth rate of IC industry revenue has stagnated. Thus, the industrial environment looks for higher efficiency on design optimization, automation, and innovation for cost reduction and performance improvement.

On the time scale of half a century, the challenges and opportunities we face today are actually the extension of, and in the same context as what we encountered before in the perspective of semiconductor history. Although we have faced many nontrivial chal-



(a) Semiconductor sales revenue worldwide from 1987 to 2018 (in billion U.S. dollars) [164].

(b) Global semiconductor industry revenue from integrated circuits from 2009 to 2018 (in billion U.S. dollars) [154].

Figure 1.1: Illustrations of semiconductor industry market size for the past decades. The asterisk marks at year indices indicate the value by forecasting.

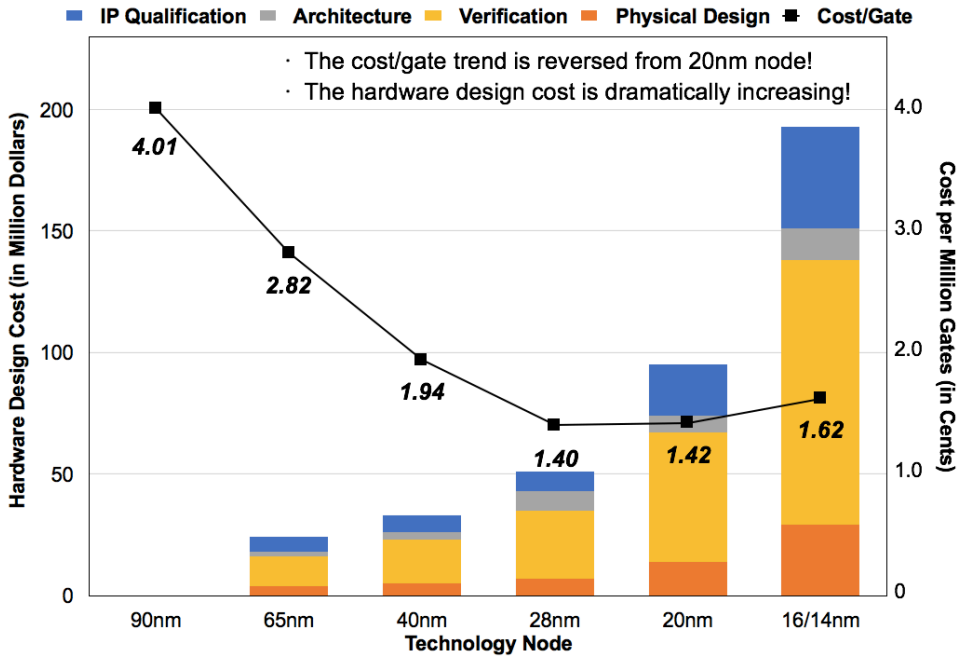


Figure 1.2: Distribution of hardware design cost per technology node (bar chart) and the cost per gate trend (line) [155]. Hardware design cost at 90nm technology node is not available from the reference [155].

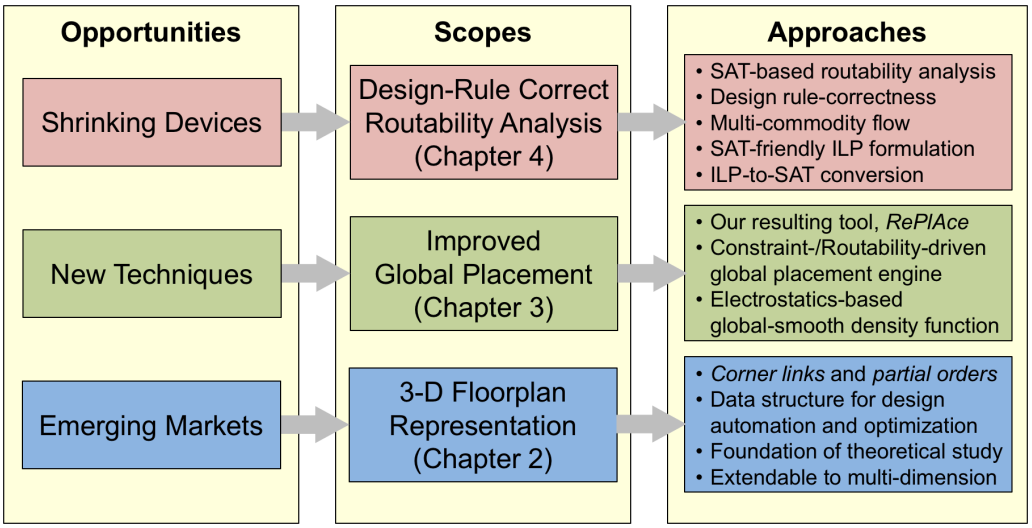


Figure 1.3: The organization of this dissertation. New 3-D floorplan representations, improved global placement engine, and SAT-based design-rule correct routability analysis.

lenges in the past, the integrated circuit industry has overcome and grown by our research and development. With this in mind, in the remaining Chapter 1, we present the current challenges on physical design and present opportunities that enable us to drive new growth engine. Then this dissertation describes new methodologies for improved physical design and its automation in three topics; (i) three-dimensional floorplan representation in Chapter 2, (ii) constraint-driven and routability-driven global placement in Chapter 3, and (iii) design rule-correct routability analysis based on Boolean satisfiability (SAT) formulation in Chapter 4.

The remainder of this chapter is organized as follows. Section 1.1 reviews researches and progresses on VLSI physical layout design. Section 1.2 introduces new opportunities on physical design with reference to the International Technology Roadmap for Semiconductors (ITRS). Section 1.3 concludes this chapter.

1.1 Researches on Physical Layout

Physical layout formulation is based on the VLSI technologies. On the other hand, physical layout enables the technologies, reduces design gaps, and extends the reach of the design capability. The academic activities have blossomed following the original

mission started by the SHARE committee half a century ago. The SHARE committee was formed in 1955 with the purpose of "Share to Help Avoid Redundant Effort". In 1964, the committee organized the first design automation conference (DAC) to provide a medium whereby people can interchange ideas, techniques, experience, and even specific programs on a regular basis.[1] In this section, we summarize researches and progresses on physical layout designs.

1.1.1 Back-Board Ordering

The placement paper published at the first Design Automation Conference (DAC 1964) [130] is about back board ordering. At the very early days of physical layout design, most design complexity occurs at board and system levels. Placement of back boards was formulated as a one-dimensional ordering problem. We arrange circuit boards on a backplane in a linear order to minimize the wiring requirement of the backplane. Clearly, the quality of the placement causes a direct impact on the cost and performance of the system. Thus, the importance of physical layout optimization/automation is recognized in the early time of circuit designs. In [35], the author devised graphs to represent the placement process and thus used graph theory and Dijkstra algorithm to search for the optimal solutions. This work is one of the early attempts to use graph theory and mathematical algorithms for physical layout design automation.

1.1.2 Two-Dimensional Physical Layout Design

For IC chip designs, we treat the placement as a two-dimensional (2-D) problem. In the transition between 1970s and 1980s, as the on-die circuit complexity increased, the need of placement tools became evident: layout designers could no longer handle hundreds of objects to produce high quality results in short turnaround. Early works in 1970s [36] [37] [38] [39] took advantage of previous works on one-dimensional ordering to improve the two-dimensional placement by rows or by columns iteratively, then the authors performed a two-dimensional placement with a tree search approach. After the

[1] The first three DACs (i.e., DAC'64-DAC'66) were called as the SHARE design automation workshop. Later, International Conference on Computer-Aided Design was started in 1981 and then ISPD in 1997 after several years of workshop as a predecessor.

investigation, an in-house place-and-route (P&R) system (namely *LAMBDA*) [39] was announced as an industrial physical layout design tool.

As the circuit complexity kept growing, we observed several breakthroughs in placement algorithms to tackle the design gaps. With interconnect dominance, placement solution plays a significant role on system performance. IC designers allocate heavy efforts to improve the placement quality. We have seen the innovations via netlist/graph partitioning methods, combinatorial algorithms, and hierarchical approaches. In the 1980s when the placement sizes are less than a hundred thousand components, annealing-based methods produced excellent results [109]. However, as the problem sizes grew, the scalability of the iterative methods became a bottleneck. Partitioning-based placement methods could reduce the complexity using hierarchical methodologies at the expenses of quality loss due to suboptimal partitioning results.

As the placement problem size approached million components, major placement tools started with analytical algorithms using quadratic or nonlinear optimization. The analytical methods were proposed in the late 1970s [6] [51] [104] [133]. Throughout the years, many innovations made the method efficient and effective.

- The sparsity of the circuit was exploited with a quadratic formulation [22].
- The linear wire length metric was approximated with nonlinear equations.
- The cell density was enforced by partitioning, pulling anchors or local repulsive forces [15] [19] [22] [62] [69] [72] [123].
- Recently in ePlace [80] [81], the density was treated as electronic charges. The balance of the electrostatic system was modelled as classic Poisson equations. The global repulsive force could then be derived using fast Fourier transform with $O(nlogn)$ complexity where n is the size of the problem.
- At the architecture level, floorplanning is a useful tool for physical layout design automation [44] [91] [95] [102] [118] [134].

With the highly competitive environments discussed above, the semiconductor IC market demand us more efforts and innovations to develop new design methodologies and to improve IC's performance. The role of physical design (PD) is to execute the physical

implementation flow, which carries the system design process. Physical design is one corner stone of a reliable, predictable implementation fabric [58] that enables system-level signoff. For a "modern" IC design, PD engineers must consider all design aspects together with various nontrivial design goals.

Due to the important role in the IC implementation steps, placement is one crucial research subject and we have many topics to further improve the solution quality. Thus, we should push the edge to further improve the solution quality and reduce the design gaps. In [5] [88], the authors summarize researches and progresses on placement and suggest directions that require intensive studies. As the authors of [5] [88] commonly mentioned, new approaches for performance-driven placement are intensively studied and should be studied in addition to the traditional placement metrics. We have seen in placement articles, the key words such as timing driven [87], routability driven [41], clock aware, datapath aware, signal integrity aware, power aware, etc. Recently, low-power design became one target objective for IC designs [5] [159]. The effort is to reduce packaging cost, increase battery life, and achieve high performance without thermal stress. Still, to produce low power design with tight constraints is nontrivial. In addition, mixed-size placement needs further improvement [88] [138]. In mixed-size designs, large modules can block routing space and enforce detours on critical interconnect. Thus, manual intervention is still wanted to place, refine, and align large modules. Additionally, tighter vertical integration in the design flow is important for the system performance. For example, interactions and convergences among physical design procedures frequently occur to reduce cost and improve solution quality: interactions between placement and gate sizing, buffer insertion, design for manufacturing; interface with logic synthesis, clock tree synthesis (CTS) and engineering change order (ECO). Thus, PD designers must consider more design aspects than before. Separately, a recent publication [61] gives a couple of interesting data points/correlations with respect to design automation research and design automation research outputs (papers, patents, and EDA companies).

1.1.3 Multi-Dimensional Physical Layout Design: 3-D, Potentially 4-D and Beyond

The extension to three-dimensional placement was initiated for the mapping of the dynamic FPGA (field-programmable gate array). Physical layout becomes a compiler tool for the high performance reconfigurable computing. Since the same space can be time shared by various functions, time domain becomes the third dimension for the placement. In this case, the turnaround time for the placement became even more critical since compilation is part of user experiences.

Recently, as three-dimensional IC technologies emerged as a promising option for the "more-than-Moore" strategy, three-dimensional layout became one key piece of the puzzle to enable the technology. We have found that with the addition of the third dimension, the floorplan topologies are much more complicated and the problem appears intriguing [103]. On top of three-dimensional physical space, we can add time as an extra dimension to make layout four-dimensional. Potentially, the number of dimensions can increase with extra factors such as energy, thermal, security and etc.

For three-dimensional placement, physical design engineers must address the challenging issues such as 3-D IC thermal distribution, floorplanning, clock tree synthesis, power distribution network, stacking methods by using through-silicon via (TSV) or monolithic inter-tier via (MIV), etc.

1.1.4 Benchmarks

The release of the benchmarks enables the researchers to measure the algorithms on a common platform, and makes the research relevant to practice. Therefore, many innovations can be incorporated into EDA tools for applications in a faster pace. Table 1.1 lists classic and modern placement benchmark suites while Figure 1.4 depicts the trend. As shown in Figure 1.4, the number of instances saturates after 2005, but additional design features, e.g., density and routability, contribute to design complexity. In the meanwhile, the saturated circuit size calls a necessity of benchmarks with even larger number of instances since the size of circuits does matter on the choice of methods and quality of results, e.g., using distributed computation and memory management.

Table 1.1: Placement benchmark suites, from classic to modern benchmarks.

Benchmark	**Description**
Steinberg [115]	Steinberg back-board placement
Illiac IV [116]	Board-level design for supercomputer
MCNC [13]	General purpose benchmarks for design automation
ISPD98 [3]	Physical design applications, e.g., partitioning and placement
ISPD-2005 [93]	Placement (also applicable to floorplanning and routing)
ISPD-2006 [92]	Placement with target density per benchmark
MMS [138]	Large-scale modern mixed-size (MMS) placement
ISPD-2011 [120]	(Global) Routability-driven placement
DAC-2012 [121]	(Global) Routability-driven placement
ICCAD-2012 [122]	Design hierarchy aware (global) routability-driven placement
ICCAD-2013 [71]	Placement finishing – detailed placement and legalization
ISPD-2014 [146]	Detailed routing-driven placement
ICCAD-2014 [68]	Incremental timing-driven placement
ISPD-2015 [11]	Blockage-aware detailed routing-driven placement
ICCAD-2015 [67]	Incremental timing-driven placement
ISPD-2016 [139]	Routability-driven FPGA placement
ISPD-2017 [158]	Clock-aware FPGA placement

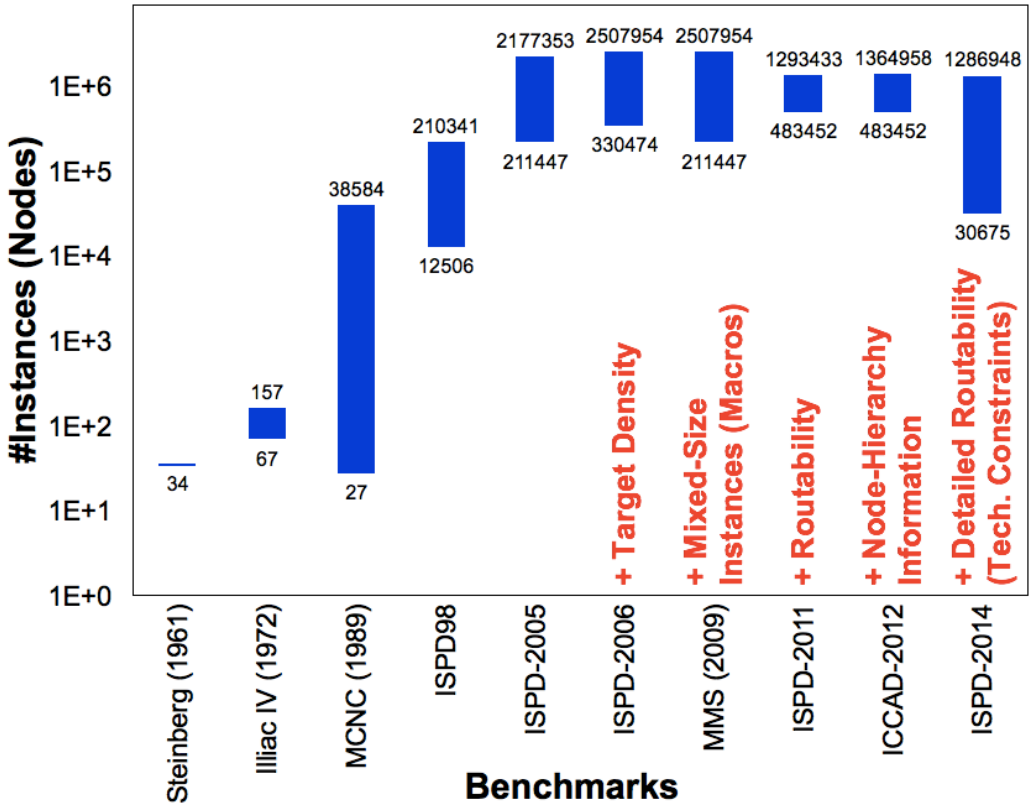


Figure 1.4: The benchmarks plotted with ranges of instances (displayed in log scale). Benchmarks are selectively chosen from Table 1.1. The minimum and maximum instances are displayed at the bottom and the top of each bar, respectively.

1.2 New Opportunities On Physical Design

In this section, we introduce new opportunities on physical design in three aspects: (i) the international technology roadmap for semiconductors (ITRS) shows that technology is going through divergence and evolution, which calls for new physical layout researches; (ii) paradigm-shifting design methodologies and innovative algorithmic techniques can improve design automation tools, and (iii) the semiconductor market and economy forecast show potential returns for further investments from the government and the industry.

1.2.1 Physical Design and ITRS

The latest ITRS [159] report suggests numerous challenging research topics. Figure 1.5 illustrates the transistor structures for logic devices. For 10nm technology, the transistor structure use finFET and the technology node will advance to 7nm for mass production soon. As shown in Figure 1.6, the extensions to the existing FDSOI and finFET will sustain for two or three technology nodes until 2020 [159]. Beyond 2020 a transition to gate-all-around and potentially to vertical nanowires devices will be needed, providing new research topics.

Design of the routable and manufacturable layout for integrated circuits (ICs) has been more and more challenging as technology nodes are continuously advanced to sub-10*nm* [159]. One of the major difficulties is caused by the resolution limitations of optical lithography, using 193*i* wavelength [40] [110]. Multi-patterning techniques such as LELE (litho-etch-litho-etch), SADP and SAQP (self-aligned double and quadruple patterning) enable successful development of 10*nm* and sub-10*nm* technology nodes for foundries. However, they also induce complex conditional design rules for manufacturability which introduce new huddles for IC design.

Securing pin accessibility of IC components has become a critical bottleneck during detailed routing, due to less number of routing tracks and increasing pin density [110] [136]. FinFET device with smaller pin geometry makes the pin accessibility problem even harder [1]. The number of routing tracks in a single placement row has been

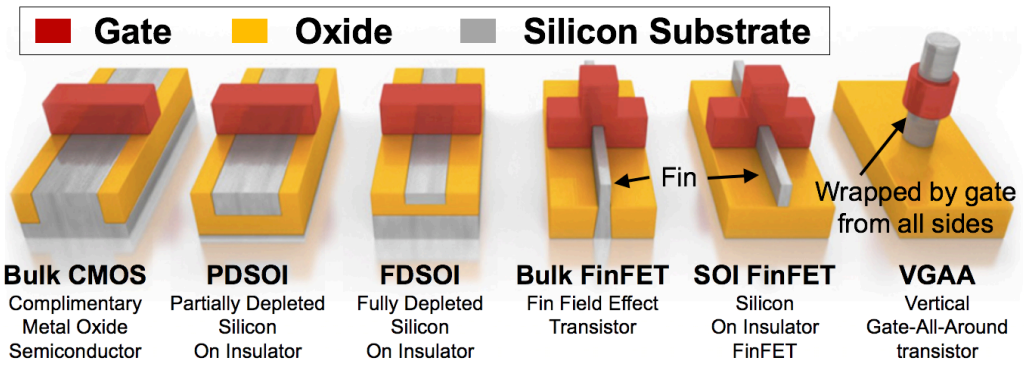


Figure 1.5: Shrink scenarios for logic devices [160]. In the current market (January 2017), the state-of-the-art transistor structure is finFET with 10nm technology.

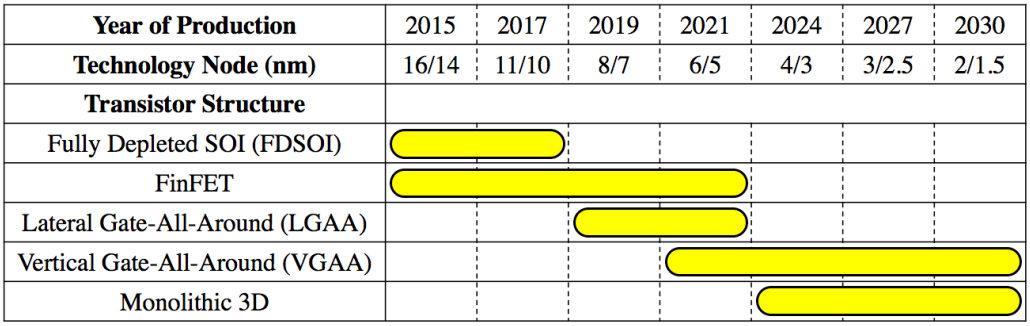


Figure 1.6: Transistor structure roadmap [159].

reduced from 12 tracks to nine/eight/seven tracks [64], and even five-track cell library with one fin is recently announced [8]. In the meanwhile, scaling metal pitch is lagged behind scaling device pitch, causing severe complications of interconnection induced by decrease of the valid access points for pins [151]. Consequently, the detailed routing step easily takes days of turnaround time (TAT), but a "successful" routing nonetheless is not guaranteed. Thus, we need a new design tool with fast TAT in analyzing the feasibility of the given layout architecture, e.g., design rules and patterns of pin assignment.

In Chapter 4, we propose a novel framework that efficiently identifies the conditional design rule-correct routability through well-organized ILP (integer linear programming) and SAT (Boolean satisfiability) formulation. Due to less number of pin-access points and dense pins for sub-10*nm* IC designs, the growing mismatch between global-route congestion map and detailed-route design rule check (DRC) violations may endanger the on-time tapeout by demanding too many manual layout revisions [17]. The absence of fast and precise routability analysis tool at the early physical design stage (e.g., before routing) exposes the entire IC design project to high-risk unpredictability. Our resulting framework offers an early "go/no-go" decision opportunity for the remaining PD procedure, based on the precise routability assessment.

1.2.2 New Techniques

Since designing ICs has been much more sophisticated and complicated, we need new techniques to solve many nontrivial challenges. IC layout design quality directly impacts overall design quality of results with respect to timing closure, die utilization,

routability, and design turnaround time; these in turn affect the classic metrics of operating frequency, yield, power consumption and cost. Thus, the industrial environment looks for higher efficiency on design optimization, automation, and innovation for better performance and cost reduction.

Emerging algorithmic techniques from outside fields also provide potential methods to solve some physical design bottlenecks. Attempts to apply artificial intelligence and deep learning techniques present progresses to predict potential congestion during placement and to detect data paths from netlist or design-hierarchy information. Massively parallel computing gives new chances to try distributed algorithms which can drastically improve our computation and memory capacity. We remain the adaptation of the paradigm-shifting new techniques into the physical design as the future works, but we firmly believe that these new techniques will introduce new breakthroughs to us, opening tremendous opportunities.

In Chapter 3, we describe our constraint- and routability-driven global placement engine, *RePlAce*. Placement solution quality directly impacts overall IC design quality as placement is a fundamental, critical step in the physical design (PD) of integrated circuits (ICs) [60]. Despite significant improvement in placement algorithms over the past decades [89], efficient and effective placement remains a challenging issue [5], e.g., routable placement solution. The placement process must provide a routable solution to the router. RePlAce is a flat, nonlinear analytical global placement engine with electrostatics-based global-smooth density cost function, addressing routing congestion as well as classical design goals with analogy of charge and electrical potential distribution. RePlAce is the first work to achieve superior solution quality across all the ISPD-2005, ISPD-2006, MMS, DAC-2012 and ICCAD-2012 benchmark suites with a single placement engine.

1.2.3 New Markets for IC Design

By virtue of physical designers' efforts, the performance of ICs has been extremely advanced, the overall industry has been profitable, and the global semiconductor market size has shown an upward climbing for the past decades. Consequently, the highly advanced ICs enable the recent advent of the fourth industrial revolution, evolving al-

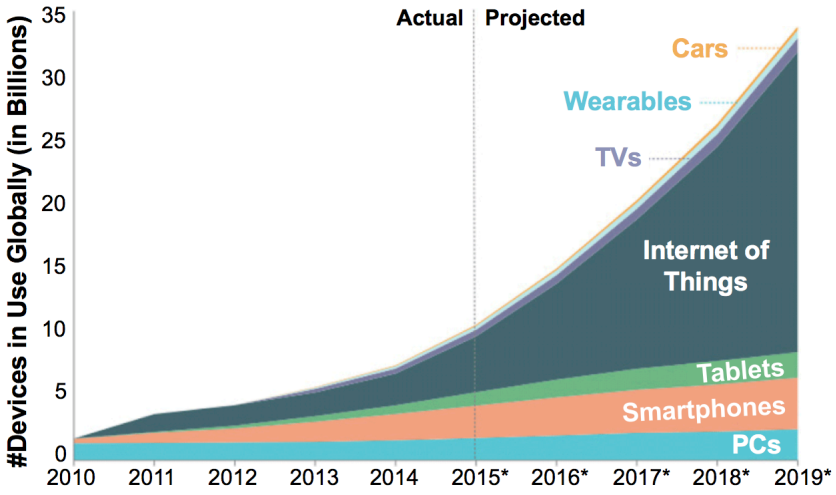


Figure 1.7: Internet of Things (IoT): The explosively growing Internet device market [119].

most every industry, e.g., artificial intelligence (AI), the Internet of things robotics (IoT), biotechnology, autonomous vehicles, etc.

Historically, the EDA industry grew from solving new design problems and the industry grew by expanding the market demand. From the mid-90s, the Internet revolution has led the skyrocketing growth of the information technology. The advancement of semiconductor enabled portable and powerful devices with more functionality. Eventually, billions of electronic devices will be connected through the Internet (so called the Internet of Things (IoT), Figure 1.7). The trend inevitably results in "small quantity batch production" of ICs ensuring low power application, by very cost-effective design and manufacturing. The new market will enable a new growth engine to our industry with new opportunities.

In Chapter 2, we propose new 3-D IC floorplan representations, *corner links*, *four trees*, and *partial order*, enhancing 3-D IC physical design automation. As the market demand smaller footprint/wirelength and less power consumption, but better performance and more functionalities, we need a new IC design methodology to deliver the high-performance ICs. While "Moore's Law" and "Dennard Scaling" have shown the correction of the slowing down, three-dimensional (3-D) ICs offer a potential breakthrough to enable a paradigm-shift strategy, called "more than Moore", with novel features and

advantages over the conventional 2-D process technology by the nature of 3-D IC fabrication. With the three-dimensional (3-D) interconnections, 3-D IC provides substantial wirelength reduction and massive amount of bandwidth, leading performance improvement to overcome many of the nontrivial challenges of the semiconductor industry. Moreover, 3-D IC integration enables to stack disparate technologies with various functionalities into a single system-in-package (SiP), introducing "true 3-D IC" design.

1.3 Conclusion

In this chapter, we briefly review the previous paths and challenges in physical layout design field. Then, we introduce and summarize new opportunities of physical design related researches. These new opportunities strongly motivate the researches in this dissertation.

1.4 Acknowledgments

Chapter 1 contains a reprint of Ilgweon Kang and Chung-Kuan Cheng, "Physical Layout after Half a Century: From Back-Board Ordering to Multi-Dimensional Placement and Beyond", *Proc. ACM/IEEE International Symposium on Physical Design*, 2017. The dissertation author is the primary author of the paper.

I would like to thank my coauthor Chung-Kuan Cheng.

Chapter 2

3-D Floorplan Representations by Using Corner Links and Partial Order

Three-dimensional integrated circuit (3-D IC) technology offers a potential breakthrough to enable a paradigm-shift strategy, called "more than Moore", with novel features and advantages over the conventional 2-D process technology. With the three-dimensional interconnections, 3-D IC provides substantial wirelength reduction and a massive amount of bandwidth, which gives significant performance improvement to overcome many of the nontrivial challenges of semiconductor industry. Moreover, 3-D integration technology enables stacking disparate technologies with various functionalities into a single system-in-package (SiP), introducing "true 3-D IC" design.

As the first physical design (PD) step, floorplanning takes a crucial role to determine IC's overall design qualities such as footprint area, timing closure, power distribution, thermal management, etc. However, lack of efficient 3-D floorplanning algorithms that implement advantages of 3-D integration is a critical bottleneck for PD automation of 3-D IC design and implementation. 3-D floorplanning is a well-known NP-hard problem, and most of 3-D floorplan algorithms rely on heuristics and iterative improvements. Thus, developing efficient 3-D IC floorplan representations is important since floorplan representation provides the foundation of data structure to search the solution space for

3-D IC floorplanning. A well-defined floorplan representation provides an organized and cost-effective methodology to design high-performance 3-D IC.

We propose a new 3-D IC floorplan representation methodology using *corner links* and *partial order*. Given a fixed number of cuboidal blocks and their volume, algorithmic 3-D floorplan representations topologically describe orientations and physical positions of each block relative to the origin in the 3-D floorplan space. In this chapter, (1) we introduce our novel 3-D IC floorplan representation, called *corner links*, (2) we analyze the equivalence relation between the corner links and the corresponding *partial order* representations, and (3) we discuss several key properties of the corner links and partial order representations. We demonstrate that the corner links representation can be reduced to their corresponding partial order representation. Also, the corner links representation for the non-degenerate 3-D mosaic floorplan can be equivalently expressed by the four tree representation. The partial order representation defines the topological structure of the 3-D mosaic floorplan with three transitive closure graphs for each direction and captures all cutting planes in the floorplan in the order of their respective directions. If the partial order representation describes relations between all pairs of blocks in the 3-D floorplan, then the floorplan is a valid floorplan. We show that the partial order representation can restore the absolute coordinates of all blocks in the 3-D mosaic floorplan by using the given physical dimensions of blocks.

2.1 Introduction

Three-dimensional (3-D) integrated circuit (IC) technology (as shown in Figure 2.1(a)) offers a potential breakthrough with novel features and advantages over the conventional 2-D integration technology. 3-D IC technology provides strong momentum to overcome many of nontrivial challenges that arise from the semiconductor industry's relentless push into the deep nanoscale regime [74]. Recently, a feature editor of *Nature* announced the paradigm shift from shrinking the devices to a new strategy called "more than Moore" [124]. The author [124] declares that 3-D IC technology is a potential option to extend the trend. With the three-dimensional interconnections, wirelength can be substantially reduced and massive amounts of bandwidth can be achieved between device layers without incurring the usual latency penalties. Also 3-D IC technology introduces

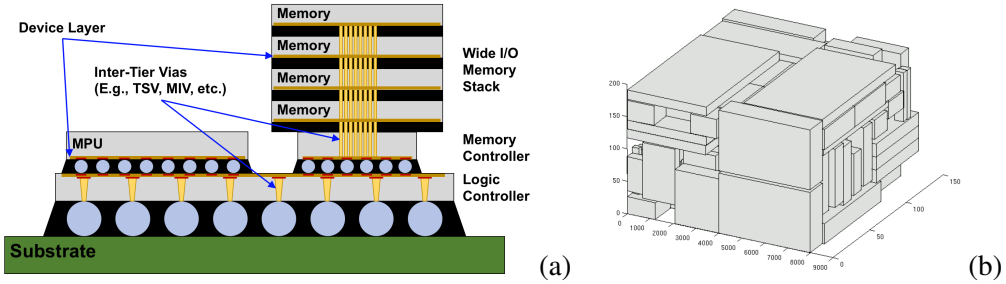


Figure 2.1: An example of 3-D IC. (a) A 3-D IC package, re-illustrated based on the original figure in [165]. (b) A general 3-D floorplan to map a design into physical space of hardware [129].

new architectures that can have much better performance for the future applications. For example, 3-D IC technology enables us to stack and integrate disparate technologies into a single system-in-package (SiP) implementation. Fabrication technologies to specific functions such as RF circuits, memories, or optoelectronic devices are often incompatible with the normal IC integration processes for high-performance logic devices. 3-D IC technology suggests a flexible and promising way to include device layers with distinct functionalities, toward "true 3-D IC".

A number of studies have validated the benefits of 3-D IC integration technology, and have explored the design space for 3-D architectures and physical implementations, which maximize the advantages of 3-D ICs. Bernstein et al. [10] confirm the benefits of 3-D integration in the scope of architecture and performance. In [132], the authors demonstrate that a multicore processor chip design could reduce the number of interconnections between critical intercore components by an order of magnitude through 3-D IC implementation. Hybrid memory cube (HMC) consortium [156] is another example of efforts from industrial leaders to drive 3-D ICs into mainstream production.

Despite the recent emphases on the necessity of 3-D IC designs and implementations, a multitude of challenges has so far obstructed large-scale transition from the classical 2-D ICs to stacked 3-D ICs [73]. Knechtel and Lienig discuss the most relevant aspects of automating the physical design (PD) process for 3-D ICs by highlighting how 3-D IC design becomes increasingly difficult and demanding as compared to well-engineered design automation for 2-D ICs. Major design challenges for 3-D ICs include 3-D IC-specific challenges such as 3-D fabrication technologies, 3-D architecture explo-

ration, system-level interconnection design, 3-D stacking-aware partitioning, etc., as well as traditional challenges such as placement, clock-tree synthesis, thermal management, reliability, power distribution, etc. For a past decade, 3-D IC design automation has been intensively studied by both industry and academia [16] [21] [25] [34] [57] [82] [101] [107] and not yet sufficient. Therefore, more sophisticated and practical solutions are urgently required to fully exploit the benefits of 3-D IC technology. For example, for power distribution, IC designers encounter many potential design choices of the packaging technologies such as on-chip and off-chip voltage regulators, decoupling capacitors, number of dies, wire pitches and allocation of vertical vias (i.e., through-silicon via (TSV) or monolithic inter-tier via (MIV)) for millions of nodes [49] [53] [111] [149]. The difficulties of 3-D IC designs and implementations call for powerful and efficient design automation algorithms to search and optimize the 3-D IC design space.

In this chapter, we focus on the 3-D IC floorplanning representations to enable effective 3-D IC design automation. Floorplanning is the first step in the physical design for VLSI circuits. Thus, the solution quality of floorplanning directly impacts overall IC design quality with respect to footprint area, die utilization, interconnection, routing congestion, power delivery network, timing closure, etc. With this in mind, the spatial and topological structuring of non-overlapping blocks (or block packing) is an important theoretical research topic for the IC floorplanning. The ultimate goal of having good spatial structure is to optimize the performance of ICs in terms of timing closure, die utilization, power consumption, production cost, design flow, etc.

Floorplan representation is crucial for IC designers as the foundation of data structure, providing an early perspective of the entire chip-level solution quality. Without an efficient and complete methodology for the floorplan representation algorithms, designing efficient floorplan becomes extremely hard due to the lack of computational tools for IC designers to study the IC floorplanning solution space. While conventional 2-D floorplanning researches have published a large amount of literature and prominent results, the extension to 3-D IC floorplanning remains open with many challenges to explore. In this work, we introduce the new 3-D IC floorplan representation by using corner links and partial order. We also propose "four trees" representation which equivalently expresses the corner links of the 3-D floorplan. We describe several key properties of the corner links and partial order representations. The main contributions are as follows.

- We introduce a new 3-D floorplan representation methodology by using **corner links** and **partial order** representations.
  - We define a *corner link* as a set of $1/8$ corners (i.e., vertices) that intersect at the same coordinate, and belong to the adjacent blocks next to each other in X, Y, Z, or diagonal direction.
  - The partial order defines the topology of the floorplan with three transitive closure graphs per each direction. There are two partial-order-related representations; (i) *face partial order* and (ii) *block partial order*.
- We analyze the relations between the corner links representation and the partial order representation.
  - The corner links representation can be reduced to the partial order representation.
  - For the non-degenerate 3-D mosaic floorplan, the corner links can be equivalently expressed by a set of *four trees* representation.
  - If the partial order representation describes relations of all pairs of blocks in the 3-D floorplan, then the partial order representation produces the valid floorplan.
  - The partial order representation yields all cutting planes in the 3-D floorplan, in sorted order by their respective directions.
- We present two algorithms for the following transformations:
  - From the corner links representation to the partial order representation.
  - From the partial order representation to the absolute coordinates of the entire blocks in the floorplan.

The remainder of this chapter is organized as follows. Section 2.2 reviews general foundations of classes of floorplans and floorplan representations. Section 2.3 introduces our new 3-D IC floorplan representation, i.e., corner links, and partial order representation for 3-D IC floorplanning. Section 2.4 demonstrates the properties and relationships of the corner links, four trees, and face and block partial orders. Then we demonstrate

these properties. Section 2.5 describes the algorithms to convert corner links to partial order, and partial order to absolute coordinates of the entire blocks in the 3-D floorplan. Section 2.6 concludes this chapter and present the future works.

2.2 Floorplan Overview

In this section, we first introduce the basis of floorplanning and the related terms such as block overlap, block adjacency, and valid floorplan. Next, we describe three floorplan classifications. In this work, we focus on a class of mosaic floorplans. Then we present the fundamental floorplan representations, and several key characteristics of the mosaic floorplan.

2.2.1 Basis of Floorplanning

In a d-dimensional (d-D) floorplanning (sometimes called as box partitioning) problem, suppose we have a d-D floorplan space P. The floorplan space P is the rectangular or block that covers the entire block components. We partition the given floorplan space P into n blocks (or more generally find an arrangement of n non-overlapping blocks within P). Each block B occupies the space spanning in an interval $[\min B^i,\ \max B^i]$ for each dimension i where B^i is the coordinates of block B in dimension i. In this work, we refer $\min B^i$ and $\max B^i$ as B^{i-} and B^{i+}, respectively. Following this convention, for 3-D floorplanning, we use symbols X, Y, and Z to represent each dimension i in the 3-D floorplan space P. Obviously, the strict inequality should hold for all blocks in P, i.e., $\min B^i < \max B^i$ (or $B^{i-} < B^{i+}$) for all dimensions i so that all the blocks have positive volume. Note that, in the later sections of this chapter, we use superscripts for indicating block's dimensional information (e.g., B^{X+}), and subscripts for indicating block's index (e.g., B_1). We define four terminologies, frequently used in this work as follows.

Interval Overlap

For two blocks B_1 and B_2, we define *interval overlap* in a dimension i if two intervals $[B_1^{i-},\ B_1^{i+}]$ (i.e., $[\min B_1^i,\ \max B_1^i]$) and $[B_2^{i-},\ B_2^{i+}]$ (i.e., $[\min B_2^i,\ \max B_2^i]$) hold

two inequalities: (i) $B_2^{i-} < B_1^{i+}$ (i.e., $\min B_2^i < \max B_1^i$) and (ii) $B_1^{i-} < B_2^{i+}$ (i.e., $\min B_1^i < \max B_2^i$). The strict inequality ensures that the overlap has positive length.

Block Overlap

We define *block overlap* of two blocks B_1 and B_2 if intervals $[B_1^{i-},\ B_1^{i+}]$ (i.e., $[\min B_1^i,\ \max B_1^i]$) and $[B_2^{i-},\ B_2^{i+}]$ (i.e., $[\min B_2^i,\ \max B_2^i]$) overlap for all dimensions i in the given d-D floorplan space P. In other words, two blocks B_1 and B_2 are not overlapped if there exists a dimension i that does not hold the interval overlap.

Block Adjacency

A pair of blocks B_1 and B_2 are *adjacent* if there exists a dimension i such that $B_1^{i-} = B_2^{i+}$ (i.e., $\min B_1^i = \max B_2^i$) or $B_2^{i-} = B_1^{i+}$ (i.e., $\min B_2^i = \max B_1^i$); and for the remaining dimension(s) j where $j \neq i$, all the intervals $[B_1^{j-},\ B_1^{j+}]$ (i.e., $[\min B_1^j,\ \max B_1^j]$) and $[B_2^{j-},\ B_2^{j+}]$ (i.e., $[\min B_2^j,\ \max B_2^j]$) are overlapped. For 3-D floorplanning, we define two blocks B_1 and B_2 are *adjacent* if these two blocks are touching the surface of one another.

Valid Floorplan

In the d-D floorplan space P, a floorplan is *valid* if any pair of blocks are not overlapped with each other.

2.2.2 Classification of Floorplanning

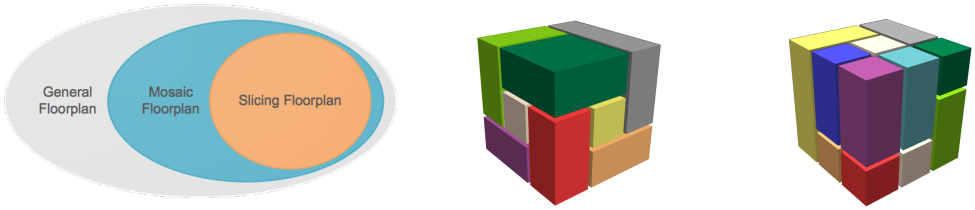
We classify floorplans into three categories, (i) general floorplan, (ii) mosaic floorplan, and (iii) slicing floorplan. Figure 2.2(a) shows the set relation of three floorplan classifications, and Figures 2.2(b) and (c) present the examples of mosaic and slicing floorplan, respectively. A set of slicing floorplans is a subset of mosaic floorplans and a set of mosaic floorplans is a subset of general floorplans. In this work, we focus on the mosaic floorplan and its representation methodology.

General Floorplan

A *general floorplan* is a collection of non-overlapping (i.e., non-intersecting) blocks, where all blocks are contained within the given floorplan space P. The general floorplan allows empty spaces within the given floorplan space P. To reduce the problem complexity, we often focus on the case of a *compact floorplan*, where all blocks are pushed toward the origin such that no block can be further shifted closer towards the origin without overlapping or moving other blocks.

Mosaic Floorplan

A set of *mosaic floorplans* is a subset of the general floorplans, obtained by partitioning the floorplan space P into multiple blocks (as shown in Figure 2.2(b)). In the mosaic floorplan, the blocks cover all available floorplan space P, indicating that there are no empty spaces within the given floorplan space P. While nonexistence of empty space is a substantial restriction compared to the general floorplans, the mosaic floorplan provides much simpler thus more efficient structure to study and analyze. This allows designers to reduce possible combinations of the solution range. Also, the mosaic floorplan can be easily extended to the general floorplan by adding empty blocks with a controlled number of gaps (i.e., holes). In addition, the mosaic floorplan gives a more general overview than the slicing floorplan so that physical design engineers can obtain deeper insights of the IC floorplanning problems. We further discuss the mosaic floorplan in Section 2.2.4.



(a) (b) (c)

Figure 2.2: Classification of floorplans. (a) Set relation chart of floorplan classifications. (b) An example of the mosaic floorplan. (c) An example of the slicing floorplan where we can partition the floorplan into single blocks by cutting blocks recursively.

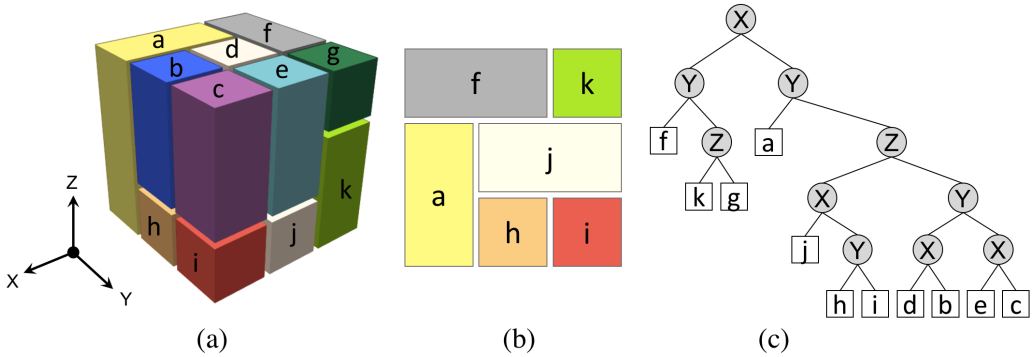


Figure 2.3: An example of 3-D slicing floorplan with 11 blocks, i.e., blocks *a*-*k*. (a) The 3-D slicing floorplan from Figure 2.2(c) and its coordinate system. (b) The bottom layer layout of the 3-D slicing floorplan. (c) The slicing tree representation of the 3-D slicing floorplan.

Slicing Floorplan

A set of *slicing floorplans* is a subset of the mosaic floorplans, obtained by recursively bi-partitioning (i.e., slicing) a block in the floorplan space P into two blocks with new boundaries perpendicular to a dimensional axis. The slicing floorplan starts with assuming the floorplan space P as a single block (as shown in Figure 2.2(c)). With this in mind, the slicing floorplan can be represented with a hierarchical tree structure (i.e., slicing tree) [20] as shown in Figure 2.3. By its nature, a slicing tree is a full binary tree. The root shows the first cut of the floorplan. Each node of the tree represents one of the three cutting directions in the 3-D floorplan space. Each internal node represents the further cuts on the partial subspace, derived through the path from the root to the node. The leaf contains each individual block.

2.2.3 Fundamental Floorplan Representations

Floorplan representation is a fundamental and important research topic in IC physical design. Many floorplanning problems for 2-D and 3-D ICs are NP-hard [128]. Most IC floorplanning algorithms are heuristic and rely on perturbations with random searches and iterative improvements. Thus, research on floorplan representations are important since floorplan representations provide a foundation of the data structures for floorplanning algorithms. Also, floorplan representations are a key component for efficient

iterative improvement procedure. Consequently, the floorplan representation methodologies expressing geometrical information of circuit blocks directly impacts complexity of floorplanning algorithms as well as overall design quality of results (QoR) with respect to footprint area, routability, timing closure, power consumption, etc.

Examples of simple floorplan representation methodologies are including floorplan specifications based on (i) absolute coordinate, (ii) corner-stitching, and (iii) partial ordering. The absolute coordinate-based floorplan specification is a naïve representation method that directly gives the physical locations of blocks. These can be used, e.g., for visualization purposes to specify the physical position and dimensions of each circuit block in a fixed space. While the absolute coordinate floorplan specification is usefully explicit, it is a high-entropy representation, which makes the design problems more difficult to work with and unsuitable for solving various optimization problems. Corner-stitching is a well-known representation for 2-D IC floorplanning [96] [97]. During floorplanning, we track all neighboring blocks through corner stitches connecting opposite corners (e.g., lower-left and upper-right corners in Figure 2.4) of every block. The stitches combined all together form the data structure for the given floorplan. In this work, our new 3-D floorplan representation methodology, i.e., corner links, is deeply inspired by the corner-stitching representation (see Section 2.3.1). The partial ordering provides the topological structure of a floorplan with transitive closure graphs per each dimension. There are two partial order-related representations; face partial order and block partial order. The details are described in Section 2.3.2.

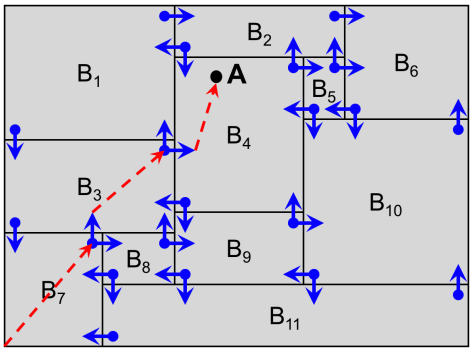


Figure 2.4: An example of corner-stitching-based 2-D floorplan representation. Each blue arrow depicts stitch, which is pointer. Red dotted arrows show the searching path for the block containing point *A* by following the pre-defined pointers.

2.2.4 Mosaic Floorplan

In this section, we discuss the key features of the mosaic floorplan. For mosaic floorplan, block sizes are flexible since the topological structure remains the same even after adjusting block sizes.

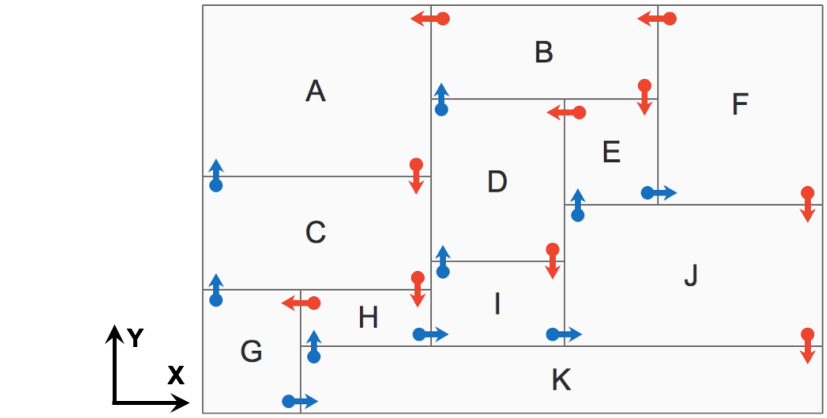
Key Features

The 2-D mosaic floorplan is well-described in [44], and the authors summarize the key features of the 2-D mosaic floorplan as follows:

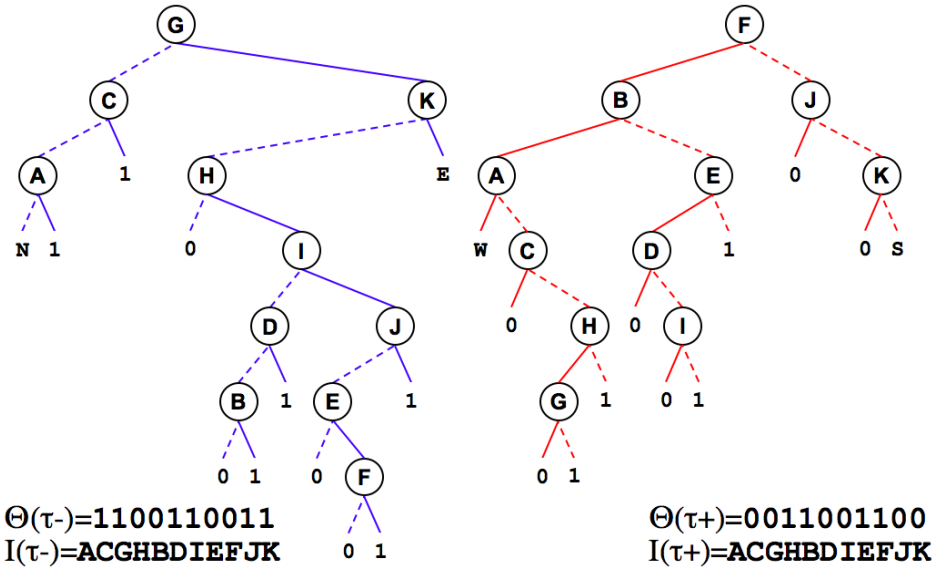
- No empty space exists within the floorplan space, i.e., each coordinate is assigned at-least one and only one block.
- The segment intersection forms a "*T-junction*", except the corners of the given floorplan space. The T-junctions (0°, 90°, 180°, 270°) are defined as the point where one end of the non-crossing segment contacts the crossing segment.
- There is no *degenerate* case where two distinct T-junctions meet at the same location. We define the *non-degenerate floorplan* as the floorplan that does not have any degenerate cases.
- The topological structure of the mosaic floorplan remains the same even after adjusting block sizes by shifting T-junction corners, unless the neighboring corners are non-degenerate.

2-D Mosaic Floorplan

The mosaic floorplan has been well-studied in the past two decades [14]. Also, many advances have been made in 2-D floorplan representations. In particular, our research on the new 3-D floorplan representation, i.e., corner links, is inspired by corner stitching [96] [97] and twin binary trees [141] [142] representations. For the 2-D mosaic floorplans, Yao et al. [141] present the precise number of mosaic floorplans with the given-fixed number of blocks. The number of distinct mosaic floorplans with n blocks is equal to the n-th Baxter number $B(n)$ [9] [23] as shown in Equation (2.1). Dulucq and Guibert [30] also prove that the exact number of distinct twin binary trees with n nodes



(a)



(b)

Figure 2.5: (a) An example of the 2-D mosaic floorplan, which has the same topological structure with the 2-D floorplan in Figure 2.4. (b) The twin binary trees for the floorplan of Figure 2.5(a). The pair of trees represents the up-right and down-left corner relations.

is equal to the Baxter number $B(n)$. Finally, the authors of [141] demonstrate a bijective mapping between the pair of twin binary trees and the 2-D mosaic floorplan.

$$B(n) = \binom{n+1}{1}^{-1} \binom{n+1}{2}^{-1} \sum_{k=1}^{n} \binom{n+1}{k-1} \binom{n+1}{k} \binom{n+1}{k+1} \tag{2.1}$$

In this work, we define the corners of each block as $1/2^d$ where d is the dimension of the given floorplan space P. Thus in a 2-D floorplan, we refer to each corner of the four corners of a block as $1/4$ corner. Note that the two edges of the corner (i.e., a non-crossing segment at a T-junction) divide the space into a 90-degree angle. Then, a straight line (i.e., a crossing segment at a T-junction) is considered as a $1/2$ corner. Therefore, except for the four boundary corners of the 2-D floorplan space P, the tip of each $1/4$ corner is always connected to either four $1/4$ corners or two $1/4$ corners and a $1/2$ corner. As we discussed in the previous Section 2.2.4, we refer to the four-$1/4$-corner-junction configuration as a degenerate case. For the 2-D mosaic floorplan, we eliminate these configurations by shifting the segment (forming a degenerate case) by a small distance, so that we have no degenerate cases in our consideration. Without degenerate cases, i.e., in the 2-D non-degenerate floorplan, the segment intersection always forms a T-junction.

In each dimension, we refer the corner that is closer to the origin as $-$ and the corner that is farther from the origin as $+$. And we denote each corner of block B as B^{xy} (or B^{xyz} in 3-D) where x, y, and z respectively indicate the relative location of the corner as $-$ or $+$ for each dimension. E.g., $B_1^{--} = B_1^{X-Y-}$, indicating that B_1's corner locating at the minimum X and Y directional coordinate, B_1^{X-} (i.e., $\min B_1^X$) and B_1^{Y-} (i.e., $\min B_1^Y$). For block A in Figure 2.5, the lower-left, upper-left, lower-right, and upper-right corners are denoted as A^{--}, A^{-+}, A^{+-}, and A^{++}, respectively. For each block in the 2-D mosaic floorplan of Figure 2.5(a), the $--$ corners are linked by the forward arrows (depicted in blue arrows) while the $++$ corners are linked by the backward arrows (depicted in red arrows).

Figure 2.5 illustrates (a) an example of 2-D mosaic floorplan which has the same topological structure with the floorplan of Figure 2.4 and (b) the corresponding twin binary trees of the floorplan in Figure 2.5(a). The pair of trees represents the up-right and down-left corner relations together. Blue and red arrows in Figure 2.5(a) indicate up-

right and down-left corner relations, and these are depicted by blue and red edges (left and right trees respectively) in Figure 2.5(b), respectively. Solid and dotted edges in Figure 2.5(b) respectively represent horizontal and vertical relations. In Figure 2.5, $\tau-$ and $\tau+$ respectively denote twin binary trees for up-right (i.e., blue) and down-left (i.e., red) corner relations. Θ is the leaf label sequence of tree ignoring the first, and I is the last bits and the node sequence of the tree from the in-order traversal. Each in-order traversal of the two trees (i.e., $I(\tau-)$ and $I(\tau+)$) produces the same sequence, $ACGHBDIEFJK$. We extend the two trees into binary leaves with label 0 at left and 1 at right. When we traverse the leaves from left to right (except the first and the last leaves), the complement relation of the binary strings is observed (i.e., 1100110011 (from tree $\tau-$) vs. 0011001100 (from tree $\tau+$)). Combined Θ and I information together, we can decode and restore the physical structure of 2-D mosaic floorplan [141].

3-D Mosaic Floorplan

Recently, 3-D floorplans have been intensively studied. Research on 3-D floorplan representation methodology include sequences of blocks to encode the topology of 3-D packing [137], representation for the 3-D slicing floorplan [20], 3-D corner block lists (CBL) [85], 3-D bonded slice-surface grid (BSG) [148], T-tree data structure for the spatial and temporal relations of blocks in 3-D space [144], graph-based topological 3-D floorplan representation (3-D transitive closure subgraph) [145], topological structure using weighted directed graph [65], and tree-based approaches and sequences to extend twin binary tree [129] [127] [33]. Fischbach et al. present a comparative study of most of the aforementioned 3-D floorplanning methods and more in [31].

As discussed in the previous Section 2.2.4, we define the corners of each block as $1/2^d$ where d is the dimension of the given d-D floorplan space. With this in mind, for the 3-D floorplan, we define a point on a vertex of a block as $1/8$ corner since the block occupies $\frac{1}{8}$ of the space at the point. Similarly, we define a point on an edge as $1/4$ corner and a point on a face as $1/2$ corner since the block occupies $\frac{1}{4}$ of the space at the point on the edge and $\frac{1}{2}$ of the space at the point on the face. The sum of the fractional corner parts at any internal junction points is always 1.

2.3 New 3-D Floorplan Representations: Corner Link and Partial Order

In this work, we develop novel 3-D floorplan representations and present their properties. In this section, we introduce the new 3-D floorplan representation methodology using *corner link*, composed of a set of $1/8$ corners (i.e., vertices) that belong to the adjacent blocks next to each other in X, Y, Z, or diagonal direction in the 3-D floorplan space. Then we connect our new 3-D floorplan representation to the *partial order* [145] that shows the topological structure of the 3-D floorplan with three transitive closure graphs for each dimension. Figure 2.6 summarizes the relations of the 3-D mosaic floorplan, corner links and four trees representations, and face and block partial orders. Each arrow describes the transformable relations through our proposed Lemmas and Theorems in Section 2.4, and Algorithms in Section 2.5.

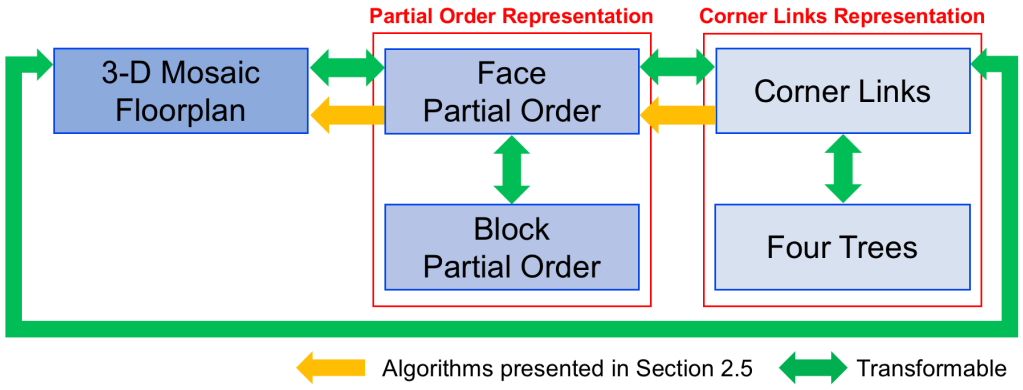


Figure 2.6: Relations of the 3-D mosaic floorplan, corner links and four trees representations, and face and block partial orders.

2.3.1 Corner Links

Corner links representation is our new representation methodology for the 3-D mosaic floorplan. Inspired by corner stitching [96] [97], 2-D mosaic floorplan representation with T-junction [44], and twin binary trees representation [141] [142], we define *corner links* as the spatial relations that describe all of the corner relations among the entire blocks in the 3-D mosaic floorplan. We define a *corner link* as a set of $1/8$ corners

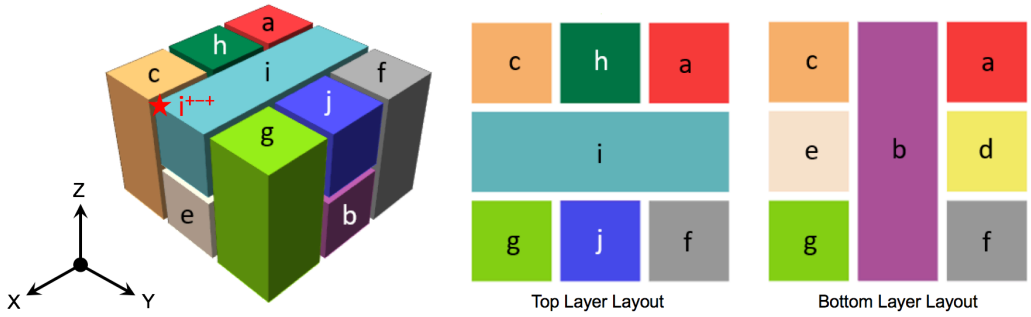


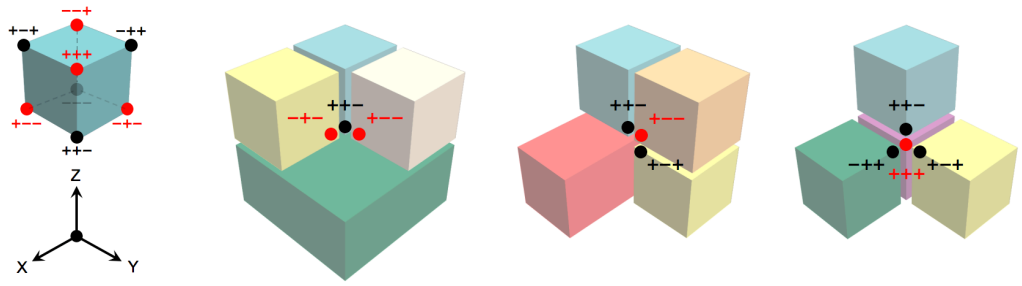
Figure 2.7: A 3-D mosaic floorplan and layouts of the top and bottom layers. There are ten blocks, a-j. A vertex of block i depicted by the red star indicates i^{+-+} corner (i^{X+Y-Z+}), meaning that the corner is located at maximum X, minimum Y, and maximum Z coordinates of block i.

(i.e., vertices) that share the same coordinate and belong to the adjacent blocks next to each other in X, Y, Z, or diagonal direction. For each corner link, we link all pairs of $1/8$ corners in the adjacent blocks together that intersect at the same point, which we call *neighboring corners*. Since we consider the 3-D mosaic floorplan, each corner link has at least two $1/8$ corners, i.e., we have at least one pair of neighboring corners in a corner link. The corner links representation describes all sets of such neighboring corners between all distinct blocks in the floorplan space P.

Figure 2.7 shows an example of the 3-D mosaic floorplan and layouts of the top and bottom layers. The 3-D floorplan space P (i.e., the outermost block that covers the entire block components for the 3-D floorplan) has the origin at the farthest point from the reader's point of view (as shown in the left-most arrows depicting X-, Y-, and Z-direction). There are ten blocks, a-j. For the 3-D floorplan space and each block B (where B is either P or a-j), we denote vertices (i.e., $1/8$ corners) of every single block as B^{xyz} where x, y, and z are either $+$ or $-$. In each dimension, we refer to the $1/8$ corner that is closer to the origin as $-$ and the $1/8$ corner that is farther from the origin as $+$, i.e., the symbols $+$ and $-$ respectively indicate the relative position of the $1/8$ corners in each block on X, Y, or Z-directional axis. For example, block i in Figure 2.7 has eight $1/8$ corners. And block i's vertex depicted by the red star means that the corner is located at maximum X- (i^{X+}, i.e., $\max i^X$), minimum Y- (i^{Y-}, i.e., $\min i^Y$), and maximum Z-coordinates (i^{Z+}, i.e., $\max i^Z$) of block i. This corner is expressed as i^{X+Y-Z+}, and simplified to i^{+-+}.

Corner i^{+-+} is linked to corner c^{+++} in Y-direction. Two corners i^{+-+} and c^{+++} are neighboring corners each other, and these two corners make a corner link, the only corner link at the location. Between two corners, the complement relation for Y-direction is observed as two corners are linked in Y-axis. Likewise, corner i^{++-} is linked to corner e^{+++} in Z-axis direction, corner e^{-++} is linked to corner j^{+--} in diagonal direction, etc. Each link determines a pair of neighboring corners. And a corner link is a set of pairs of neighboring corners on the same coordinate. A set of neighboring corners is a particular set of all corners in the 3-D floorplan that are pair-wise equal by the *corner equation*. We write corner equations to describe each pair of neighboring corners. For example, $e^{-++} = j^{+--}$ is a corner equation equivalent to $e^{X-Y+Z+} = j^{X+Y-Z-}$, meaning that $e^{X-} = j^{X+}$ (i.e., $\min e^x = \max j^x$), $e^{Y+} = j^{Y-}$ (i.e., $\max e^y = \min j^y$), and $e^{Z+} = j^{Z-}$ (i.e., $\max e^z = \min j^z$). In the 3-D mosaic floorplan of Figure 2.7, there are 36 pairs of neighboring corners and 36 sets of corner links.

Corner links of 3-D mosaic floorplan are conceptually matched to T-junctions for 2-D mosaic floorplan representation as determining the neighboring corners of blocks. 2-D mosaic floorplan can be represented by twin binary trees by using T-junctions. Similarly, corner links can build trees for 3-D mosaic floorplan representation. For 3-D floorplan, two trees are insufficient to represent all spatial relations defined by corner links. Instead, a 3-D mosaic floorplan requires four distinct trees that include all corner links information. Particularly, if the given 3-D mosaic floorplan is non-degenerate floorplan, the



(a) (b) (c) (d)

Figure 2.8: Illustrations of neighboring corners. (a) Each red dot is one of four opposite corners in 3-D floorplan, forming the root for each of four trees representation (as shown in Figure 2.9). (b)-(d) Examples of corner links.

corner links representation is equivalent to the *four trees* representation. Each of four trees is rooted at each of four opposite corners of the 3-D floorplan space P, i.e., P^{+++}, P^{--+}, P^{+--}, and P^{-+-} corners or P^{---}, P^{++-}, P^{-++}, and P^{+-+} corners. Figure 2.8 presents illustrations of neighboring corners to create four trees. In Figure 2.8(a), red dots are four opposite corners in 3-D floorplan, forming roots for four trees representation as shown in Figure 2.9. In Figure 2.8(b)-(d), red (resp. black) corners have outgoing (resp. incoming) links (i.e., edges in the four trees representation) to (resp. from) any neighboring corners in the corner link.

Figure 2.9 shows the four trees representation equivalent to the 3-D mosaic floorplan in Figure 2.7. In Figures 2.9(a)-(d), each tree is respectively rooted at the four opposite corners of the 3-D floorplan space P, i.e., (a) P^{+++}, (b) P^{--+}, (c) P^{+--}, and (d) P^{-+-}. Instead of P^{xyz}, each root of four trees is respectively denoted as g^{+++}, a^{--+}, c^{+--}, and f^{-+-} because the four opposite corners of the 3-D floorplan space P share the corners of g^{+++}, a^{--+}, c^{+--}, and f^{-+-}. Each node denotes blocks. Each edge represents each pair of neighboring corners in diagonal (black), X (red), Y (green), and Z (blue) directions. For example, there is an edge showing diagonal direction neighboring corners between blocks j and d (j^{---} and d^{+++}) in Figure 2.9(a). Since we have 36 neighboring corners in the 3-D floorplan, the total number of edges in four trees is equal to 36. On each

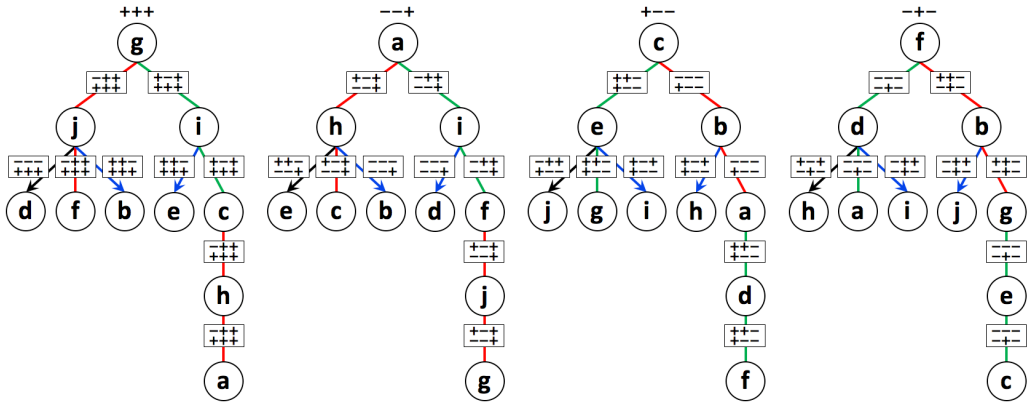


Figure 2.9: Four trees representation equivalent to the 3-D mosaic floorplan in Figure 2.7. Each of the four trees is respectively rooted at the opposite corners of the 3-D floorplan space P, i.e., (a) P^{+++}, (b) P^{--+}, (c) P^{+--}, and (d) P^{-+-}.

edge of the four trees, there is a rectangle presenting the location of two corners. The rectangle describes a pair of neighboring corners, i.e., one corner from the parent node (upper xyz symbols where x, y, z are either $+$ or $-$) and another corner from the child node (lower xyz symbols). For example, the rectangle $\begin{bmatrix} -++ \\ +++ \end{bmatrix}$ between nodes (i.e., blocks) g and j in Figure 2.9(a) represents that two corners g^{-++} and j^{+++} fabricate a pair of neighboring corners. In other words, blocks g and j are adjacent next to each other in X-axis direction and share the same coordinate at g^{-++} and j^{+++} corners. We refer the corner from the parent node (e.g., g^{-++}) and the corner from the child node (e.g., j^{+++}) as the linking and linked corners, respectively. Note that all linked neighboring corners in a single tree have the same corner locations for each block since each child node represents the linked and adjacent block to the direction from the root through its parent node. In Figure 2.9, edges depicted by arrows are stitching together the cutting plane p_3 of Figure 2.14. The cutting plane will be described in Section 2.3.2.

Figures 2.10(a) and (b) present the examples of constructing trees by traversing from the roots P^{+++} and P^{--+} (i.e., g^{+++} and a^{--+}), respectively. Red and blue arrows indicate the neighboring corners. Black dots are four opposite corners of the 3-D floorplan space P. For Figure 2.10(a) from the root g^{+++}, the linking neighboring corner g^{-++}

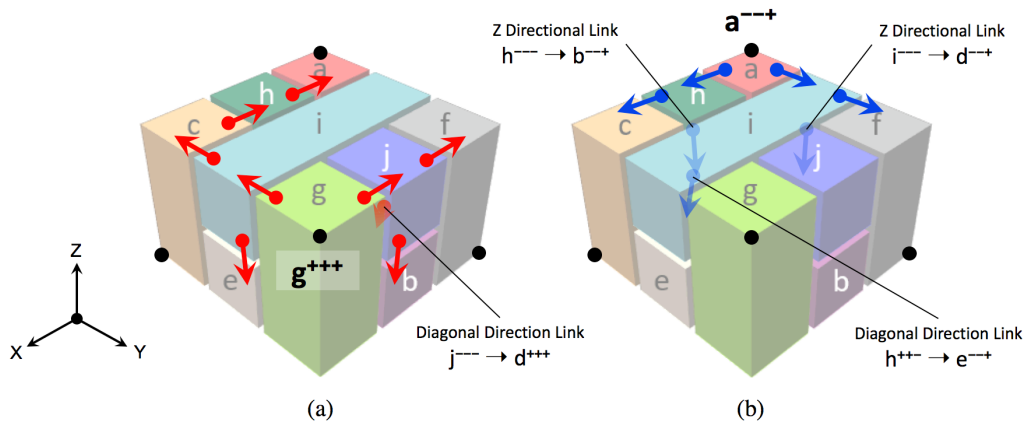


Figure 2.10: Examples of constructing trees in Figures 2.9(a) and (b). Figures (a) and (b) correspondingly illustrate traversing procedures from each root, i.e., (a) corner g^{+++} and (b) corner a^{--+}. Blocks j and h have three neighboring corners to diagonal, X, and Z directions.

links its linked neighboring corner j^{+++} in X direction. Block j has three neighboring corners from j^{-++} to f^{+++} in X direction, from j^{++-} to b^{+++} in Z direction, and from j^{---} to d^{+++} in diagonal direction. From the root, block i is linked with g^{+-+} in Y direction. Then block i links corners c^{+++} and e^{+++} in Y and Z directions, respectively. Block c links corner h^{+++}, and block h links corner a^{+++}. Figure 2.10(b) shows another example to construct tree from the root a^{--+}. Note that we must consider the links for the diagonal direction otherwise the tree cannot traverse all blocks in the 3-D floorplan. For example, the tree in Figure 2.9(a) and Figure 2.10(a) may miss block d without the diagonal-direction link, resulting in incomplete floorplan representation.

2.3.2 Partial Order

In this section, we describe the partial order representation for 3-D floorplan. The partial order defines the topological structure of the 3-D mosaic floorplan with three transitive closure graphs per each dimension [145]. In this work, we have two partial-order-related representations; (i) face partial order and (ii) block partial order representations. The face partial order is our novel representation method that is equivalent to the block partial order. Both of them can produce transitive closure graphs for each dimension.

For each dimension in the 3-D floorplan space, there exists a *face partial order* through the faces of the blocks perpendicular to the dimensional direction. We determine the face partial order by defining *face (in)equalities*, letting touching faces of adjacent blocks be equal or relating opposite faces of the same block based on their coordinate in the appropriate dimension. By taking the transitive closure graphs of each face partial order relation in each dimension, we obtain a *face partial order representation* of the 3-D floorplan. The face partial order provides complete abstraction for the topological structure of the 3-D mosaic floorplan. In this work, we determine the face partial order based on the minimum (i.e., B^{i-}, $\min B^i$) and maximum (i.e., B^{i+}, $\max B^i$) coordinate of each block B in each dimension i. I.e., the face partial order in X direction is the transitive closure graph composed of these relations by defining (i) any of two touching YZ faces of blocks as equal, and the two YZ faces of the same block as naturally ordered. Therefore, every coordinate on YZ planes in the 3-D floorplan has and belongs to the corresponding face partial order.

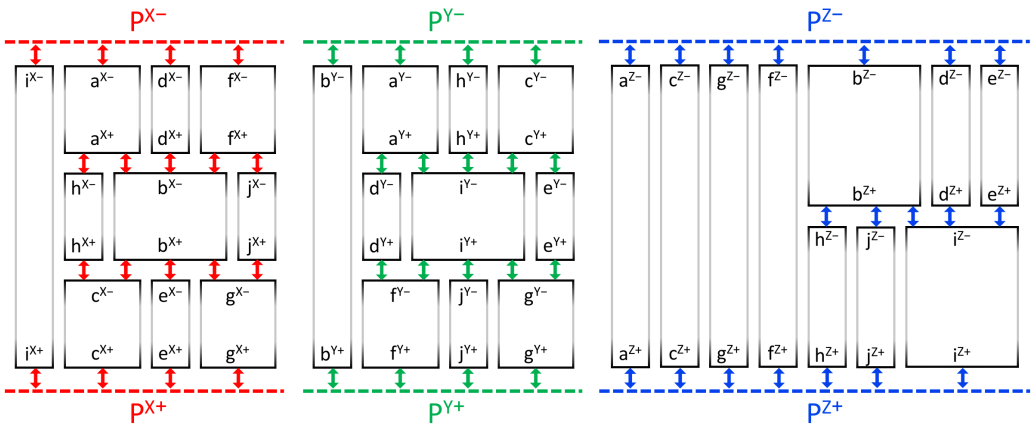
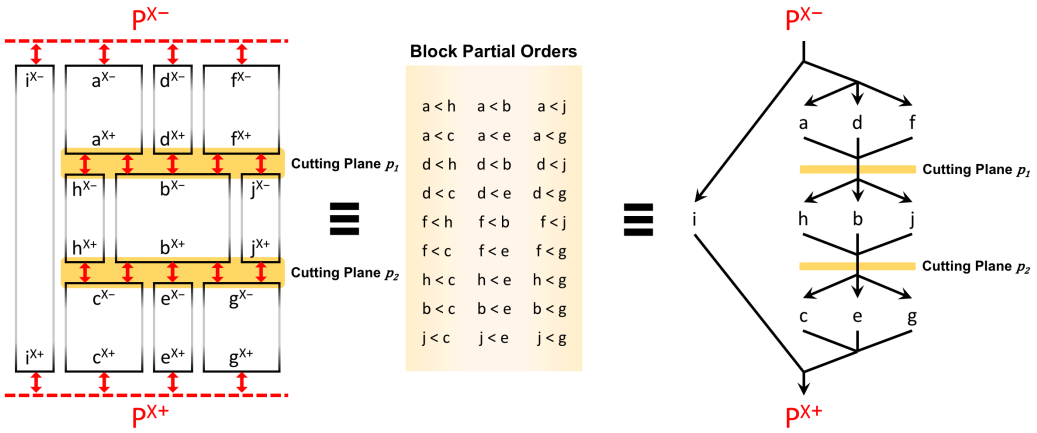


Figure 2.11: The face partial order representation for the 3-D mosaic floorplan in Figure 2.7. Figures (a), (b), and (c) present the face partial orders in X (red), Y (green), and Z (blue) directions, respectively.

Figure 2.11 is the face partial order representation for the 3-D mosaic floorplan in Figure 2.7. Figures 2.11(a)-(c) present the face partial orders in X (red), Y (green), and Z (blue) directions, respectively. The $-$ (resp. $+$) sign indicates the minimum (resp. the maximum) coordinate of blocks (i.e., the closest (resp. the farthest) faces of blocks from the origin in the dimension). For example, in Figure 2.11(a), P^{X-} (equivalent to $\min P^X$) denotes the YZ face of the 3-D floorplan space P, which has the smallest X coordinate. Likewise, a^{X-} (resp. a^{X+}) is the closest (resp. the farthest) YZ face of block a from the origin, which has the smallest (resp. the largest) X coordinate. For any point in YZ plane of block i, there are no other blocks in the X directional face partial order since block i directly touches both YZ faces of the given 3-D mosaic floorplan space P (i.e., P^{X-} and P^{X+}) . In the 3-D floorplan of Figure 2.7, there are four blocks a, d, f, and i touching P^{X-} plane (i.e., YZ face with the smallest X coordinate of the 3-D floorplan space P plane). Block a is facing block h and b, blocks h and b are commonly facing block c, and block c is facing P^{X+}. Starting with P^{X-}, finding the entire YZ-plane face partial order relations between blocks constructs the complete X-directional face partial order representation as shown in Figure 2.11(a). Similarly, we obtain Y- and Z-directional face partial order representations as shown in Figures 2.11(b)-(c).

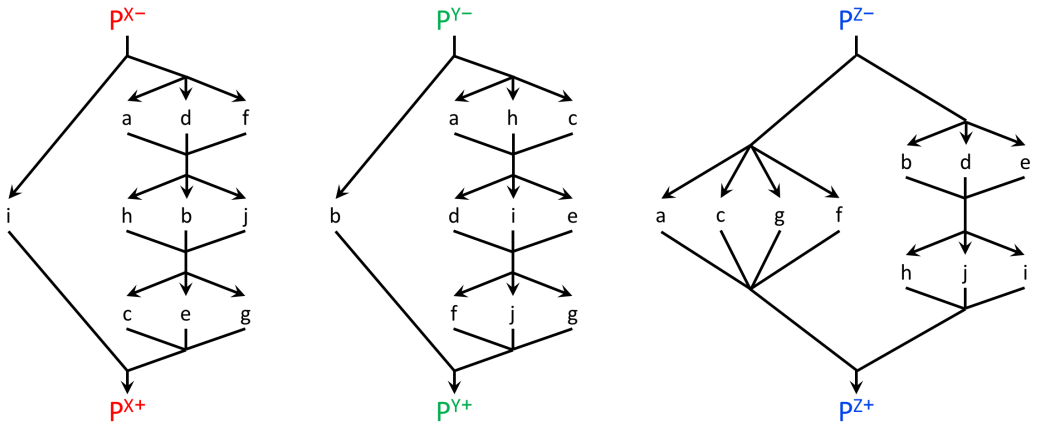
In Figure 2.11(a), we obtain the following face equations by the definition of the face partial order representation: $a^{X+} = h^{X-} = b^{X-} = d^{X+} = f^{X+} = j^{X-}$, meaning that $X+$ faces of blocks a, d, and f and $X-$ faces of blocks h, b, and j are on the same YZ plane. This YZ plane fabricates a single YZ *cutting plane*, i.e., p_1, which we define as a equivalence class through the corner equations. Similarly, we have the following relations: $h^{X+} = c^{X-} = e^{X-} = b^{X+} = j^{X+} = g^{X-}$, describing another cutting plane p_2. Intuitively, the cutting plane consists of two sets of block faces touching in a particular axis, which must have the same footprints (which we prove in Section 2.4). The cutting planes provide key concept to assemble 3-D mosaic floorplan based on the partial order representation.

We determine a *block partial order* on the blocks by defining $A > B$ (where A and B are distinct blocks in the 3-D floorplan) if each properly oriented face of A is greater than or equal to each properly oriented face of B in the face partial order of corresponding direction. For example, for the block partial order relations in X direction, $A > B$ if the minimum X-coordinate face of A (i.e., A^{X-}) is at least as large as the maximum X-coordinate face of B (i.e., B^{X+}) along with the X-directional face partial order. The block partial order and the face partial order are equivalent. And both are equivalently capable to construct the partial order representation, which abstracts the topological structure of the given 3-D floorplan with three transitive closure graphs for each dimension.



(a) (b) (c)

Figure 2.12: The equivalent relations of (a) the X-directional face partial order (from Figure 2.11(a)), (b) the X-directional block partial orders, and (c) the X-directional partial order representation.



(a) (b) (c)

Figure 2.13: The X-, Y-, and Z-directional partial order representations for the 3-D mosaic floorplan in Figure 2.7.

Figures 2.12(a)-(c) present the equivalence relations of (a) the X-directional face partial order (from Figure 2.11(a)), (b) the X-directional block partial order, and (c) the X-directional partial order representation, the X-directional transitive closure graph for the topological structure of the 3-D floorplan of Figure 2.7. Two YZ planes p_1 ($p_1 = p_1^{X-} = p_1^{X+}$) and p_2 ($p_2 = p_2^{X-} = p_2^{X+}$) fabricate two distinct YZ cutting planes (i.e., p_1 and p_2) in X direction, which we observe in Figure 2.7. For two blocks a and j, by analyzing the partial order representation, we notice that the two blocks are facing the same cutting plane p_1 although these two blocks are not directly adjacent each other. There is no corner links to stitch them together, however, the partial order representation provides intuitive overview for the cutting planes, containing key joints to assemble blocks for the given 3-D floorplan. Finally, Figures 2.13(a)-(c) respectively show the X-, Y-, and Z-directional partial order representations with three transitive closure graphs.

2.4 3-D Floorplan Representation Properties

In this section, we present several key properties for the corner links and partial order representations. Theorems and Lemmas with proofs demonstrate characteristics and relationships of the corner links, four trees, partial order representations, and their

corresponding 3-D mosaic floorplan. We show that (i) corner links representation can be reduced to partial order representation, (ii) corner links representation for 3-D non-degenerate mosaic floorplan can be equivalently expressed by four tree representation, (iii) a 3-D floorplan is a valid floorplan if the partial order representation describes relations between all pairs of blocks in the 3-D mosaic floorplan, and (iv) a partial order representation can restore the absolute coordinates of all blocks in the 3-D mosaic floorplan by using the given physical dimensions of blocks.

2.4.1 Corner Links and Partial Order Representations

The corner links representation is capable of composing the partial order representation for the mosaic floorplan, indicating that the information from the corner links representation is sufficient to abstract the topological structure of 2-D and 3-D mosaic floorplan.

Corner Links and 2-D Floorplan

Corner links representation for 2-D mosaic floorplan is equivalent to twin binary trees representation when the mosaic floorplan is non-degenerate. The twin binary trees include all T-junctions (i.e., neighboring corners) in the 2-D mosaic floorplan except at the four corners of the 2-D floorplan space P. Since every corner link is composed of a single pair of neighboring corners, twin binary trees and corner links representations are equivalent to each other for 2-D mosaic floorplan. In Figure 2.5 above, (a) and (b) respectively illustrate (a) the example of the corner links (red and blue arrows) for the 2-D floorplan, and (b) the corresponding twin binary trees based on the corner links in (a).

Corner Links to Partial Order Representation

In this section, we show that the corner links representation can be reduced to the partial order representation. As discussed in Section 2.2.4, we define the corners of each block as $1/2^d$ where d is the dimension of the floorplan space P. For 3-D floorplan, we define a vertex, edge, and face of each block as a $1/8$, $1/4$, and $1/2$ corners, respectively.

Given any internal junction point in the 3-D mosaic floorplan, the total sum of the fractional corner parts at the junction point is always 1 since mosaic floorplans do not allow any empty spaces. Consequently, there are always even number of $1/8$ corners at any internal junction point in the 3-D mosaic floorplan.

Lemma 1 (Odd Number of Neighboring Corners) *Every* $1/8$ *corner in a 3-D mosaic floorplan, other than those at the eight outermost* $1/8$ *corners of the 3-D floorplan space P, must have an odd number of neighboring corners.*

Proof:

1. The size of a single $1/8$ corner (i.e., vertex) of each block in 3-D space is $\frac{1}{8}$.
2. Except the eight outermost corners of the 3-D floorplan space P, each $1/8$ corner at a certain point is contained in $\frac{1}{4}$, $\frac{1}{2}$, or 1 of space.
3. In order to fill $\frac{1}{4}$, $\frac{1}{2}$, or 1 of the space at the point, even number of $1/8$ corners are required.
4. Therefore, each $1/8$ corner requires an odd number of $1/8$ neighboring corners to round up to $\frac{1}{4}$, $\frac{1}{2}$, or 1.

□

The XY plane between blocks b, d, e and h, j, i in Figure 2.7 is a cutting plane p_3 in Figure 2.14(a). Cutting plane p_3 is the only cutting plane in Z direction. Also, the cutting plane p_3-related edges (between neighboring corners) are depicted by blue and black arrows in Figures 2.9(a)-(d). Figures 2.14(b) and (c) present two cross-sectional views of cutting plane p_3 toward the bottom ($Z-$) direction (i.e., p_3^{Z-} in (b)) and the top ($Z+$) direction (i.e., p_3^{Z+} in (c)), respectively. Grey blocks go through cutting plane p_3 along with Z axis. By using the relevant corner links information, it is possible to show that all of non-grey-color faces in Figures 2.14(b) and (c) have the same Z-coordinates. Furthermore, we can find the symmetric difference between the set of blocks in p_3^{Z-} and the set of blocks in p_3^{Z+}. Blocks b, d, e and h, j, i from two sets make the symmetric difference. We refer the two sets as $S-$ and $S+$, respectively. Figure 2.15 shows another

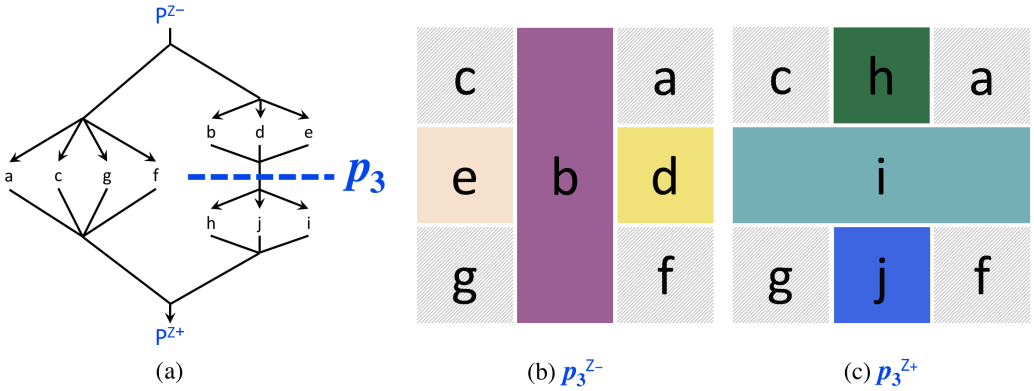


Figure 2.14: (a) Cutting plane p_3 of the 3-D floorplan example in Figure 2.7 and its Z-directional partial order in Figure 2.13(c). (b) and (c) are cross-sectional views of cutting plane p_3 toward $Z-$ (i.e., p_3^{Z-}) and $Z+$ (i.e., p_3^{Z+}) directions, respectively.

example of two cross-sectional views from cutting plane p_k, highlighting the symmetric difference by purple blocks in p_k^{Z-} (i.e., Figure (a)) and green blocks in p_k^{Z+} (i.e., Figure (b)). Note that $S-$ and $S+$ are sets of blocks in p_k^{Z-} and p_k^{Z+}, respectively.

From the above Lemma 1, we claim that the structure of corner links is a connected and acyclic directed graph. We also show that the corner links representation contains sufficient information to restore the partial order representation.

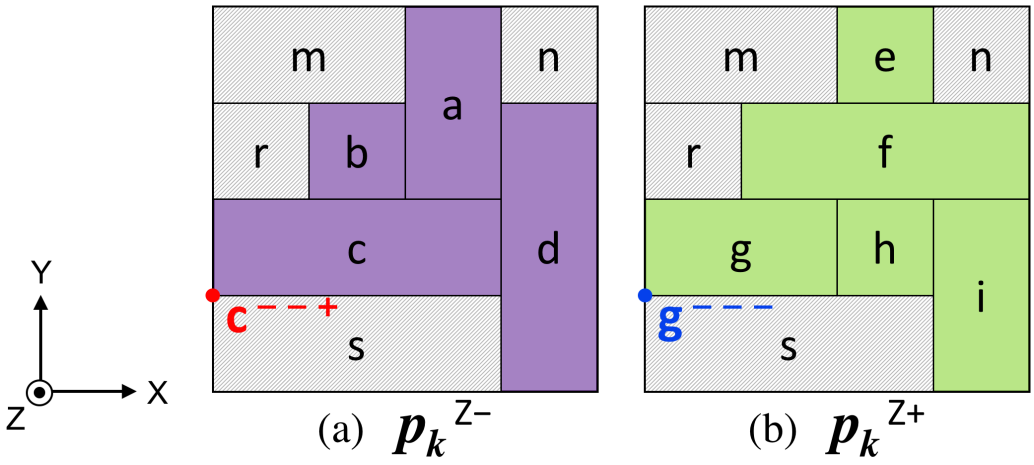


Figure 2.15: The symmetric difference for footprints across cutting plane p_k. Grey blocks go through cutting plane p_k along with Z axis. By Theorem 2.1, corner c^{--+} must have a neighboring corner, i.e., g^{---}. $S-$ and $S+$ are sets of blocks in p_k^{Z-} and p_k^{Z+}, respectively.

Theorem 2.1 (Restoring Partial Order from Corner Links) *For a 3-D mosaic floorplan, the partial order representation can be restored from the corner links representation by enumerating all blocks in p_k^{j-} and p_k^{j+} for every* ***cutting plane*** *p_k in each dimension j. Cutting plane p_k is a plane characterized by the transitive closure of face equalities obtained from faces that touch a corner, in each of the three dimensions.*

We start with adjacent blocks which have a common surface (i.e., cutting plane p_k) perpendicular to dimension j. We consider one of these blocks, B_1, looking at T that is a set of all blocks which can be shown by using the corner links (thus the corner equations) and have a face along the cutting plane p_k. Note that if any block $B_2 \in T$ shares an $1/8$ corner at a point along the cutting plane p_k with any other block B_3, then block B_3 should be in T. Also note that set T can be partitioned into two subsets according as to whether a block in T is above or below cutting plane p_k in dimension j. The two subsets give two footprints E and F, from sets of blocks in p_k^{j-} and p_k^{j+}, respectively. To prove Theorem 2.1, it is enought to show that the shapes of the two footprints are equal, i.e., $E = F$.

Proof (by contradiction):

1. Suppose that we have a 3-D mosaic floorplan. We use corner links representation to identify the cutting plane p_k. T is a set of all blocks which can be shown by using the corner links and which have a face along the cutting plane p_k.

2. Suppose that at least one block in the floorplan is not linked in the cutting plane p_k (i.e., the block is crossing the cutting plane p_k). And $S-$ and $S+$ are sets of blocks in p_k^{j-} and p_k^{j+} where j is perpendicular to the cutting plane p_k.

3. Assume footprints of $S-$ and $S+$ are not equal, i.e., $E \neq F$. Then we take a corner of the symmetric difference (e.g., a bottom-left corner c^{--+} as shown in Figure 2.15).

4. The corner (e.g., c^{--+}) lies on the cutting plane p_k^{j-} and belongs to $S-$. And $S- \in T$.

5. The corner is a $1/8$ corner of an odd number of $1/8$ corners in T as $E \neq F$ in (3).

6. However, the $1/8$ corner must be in a set of an even number of $1/8$ corners by Lemma 1. Thus, there must be block B_i that has the corner (i.e., c^{---+}) as its neighboring corner.

7. This would force $B_i \in T$ by (1), a contradiction. Therefore, $E = F$.

□

2.4.2 Corner Links and Four Trees Representation

For a non-degenerate 3-D mosaic floorplan, the corner links representation can be equivalently expressed by the four trees representation. Suppose that each corner has only one neighboring corner except at the corner of the 3-D floorplan space P. Then, the corner links representation forms trees. However, there are 3-D floorplans that have corners having more than one neighboring corner as shown in Figure 2.16. Figure 2.16 shows a special example for the non-degenerate 3-D floorplan, which has corners having more than one neighboring corner. Blocks a, b, c, and d are intersecting at the same coordinate. For 2-D floorplan, when we have corners which have more than one neighboring corner (i.e., two segments are crossing against each other), we can slide one segment to

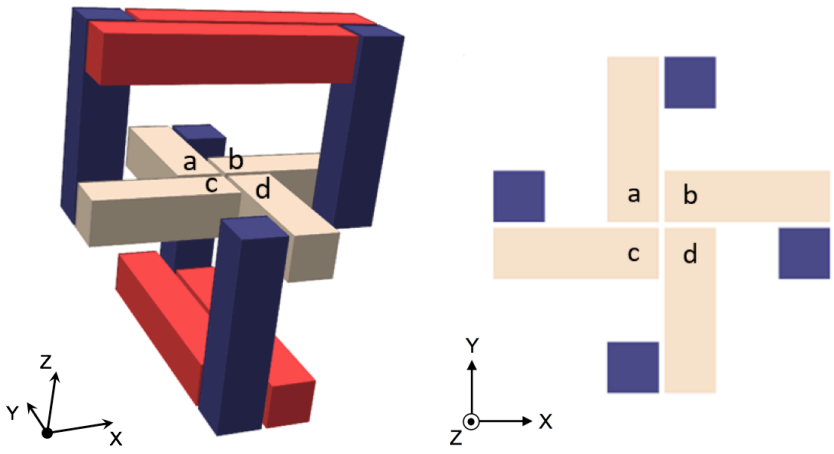


Figure 2.16: A special example for the non-degenerate 3-D floorplan, which has corners having more than one neighboring corner.

split the crossing into two T-junctions. Then the floorplan becomes the mosaic floorplan, and its corresponding twin binary trees representation gives an exact one-to-one mapping to the mosaic floorplan. However, the configuration of the 3-D floorplan in Figure 2.16 indicates that adjusting block size is inapplicable due to the alignments of red and blue blocks above and below blocks a, b, c, and d, thus the 3-D floorplan is non-degenerate.

Theorem 2.2 (Conjecture of Four Trees Representation to 3-D Floorplan) *For 3-D mosaic floorplan, four trees are sufficient for a non-degenerate floorplan representation.*

Proof:

1. Each face of a block has two corner links at two opposite corners to determine the boundary of the block.
2. By Theorem 2.1, all faces on a cutting plane are linked together.
3. Based on 1 and 2, the statement of Theorem 2.2 is true.

□

More generally, we can conjecture that every 3-D mosaic floorplan can be expressed by its corresponding four trees representation. Conversely, each four trees representation describes a single mosaic floorplan as both corner links representation and four trees representation are based on corner equations. We can extend this conjecture to higher dimensions. For d-D space, 2^{d-1} trees are sufficient for a non-degenerate d-D mosaic floorplan representation.

2.4.3 Partial Order Representation to Valid Floorplan

From the corner links, we derive an important property of the partial order representation. Based on Lemma 2, we show that the partial order representation can describe a valid 3-D floorplan. Each corner link determines relations (either descendent or sibling)

between every pair of blocks in the corner link. The partial order representation yields all cutting planes in the 3-D mosaic floorplan, in sorted order by their respective dimensions. If the partial order representation describes relations between all pairs of blocks in the 3-D floorplan, then the partial order representation produces a valid 3-D floorplan.

Lemma 2 (Partial Order based on Corner Link in 3-D Mosaic Floorplan) *Given a corner link between blocks B_1 and B_2 in a 3-D mosaic floorplan, let the locations of the two neighboring corners be $B_1^{x_1 y_1 z_1}$ and $B_2^{x_2 y_2 z_2}$ where x_1, x_2, y_1, y_2, z_1, z_2 respectively indicate the relative location of the corner as $-$ or $+$ for each dimension. If $x_1 \neq x_2$, then the two blocks B_1 and B_2 are in descendant relation in the partial order representation for X dimension. Otherwise, the two blocks are in sibling relation under the cutting plane on the $B_1^{x_1}$ side. Similarly, Y and Z dimensional relations are obtained.*

Proof:

By the definition of the corner links and the corner equations, the statement of Lemma 2 is true for all 3-D mosaic floorplans.

□

For example, from the mosaic 3-D floorplan in Figure 2.7 and the four trees representation in Figure 2.9, we have a corner link composed of two neighboring corners g^{-++} and j^{+++}, describing the relation between two blocks g and j in X dimension. By Lemma 2, block j is the parent of block g in the X-dimensional partial order. In the meantime, the two blocks are siblings under the cutting plane on the $+$ sides (i.e., $g^{Y+} = j^{Y+}$ and $g^{Z+} = j^{Z+}$) in the Y- and Z-dimensional partial order representation (as shown in Figure 2.13). Based on Lemma 2, we derive Theorem 2.3.

Theorem 2.3 (Valid 3-D Mosaic Floorplan and Partial Order Representation) *Given a valid and non-degenerate 3-D mosaic floorplan, any pair of distinct blocks are related under at least one of the partial orders.*

Proof (by enumeration):

1. The proof is by enumeration on the number of non-overlapping coordinates of the two blocks, A and B.

2. If the two blocks A and B overlap in two coordinates (e.g., assume X and Y dimensions in this proof), tracing a straight line from one to the other will provide a chain of face relations between one and the other, proving comparability in that coordinate. Note that there are no blocks overlapping in three coordinates since we consider valid 3-D floorplans.

3. If the two blocks A and B do not overlap in two coordinates (e.g., assume X and Y dimensions) but overlap in the remaining coordinate (e.g., assume Z dimension), consider the intersection with a plane (e.g., XY plane) that covers both block A and block B. Assume (without loss of generality) that A's coordinates are larger than B's coordinates in each dimension, e.g., $A^{X-} > B^{X+}$, $A^{Y-} > B^{Y+}$. Then, we consider the intersecting XY plane as a 2-D floorplan. We produce a chain of blocks $A_0, A_1, A_2, \ldots, A_n$ where $A = A_0$ and block A_{i+1} has an $1/4$ corner that shares the corner with block A_i's $--$ corner (i.e., A_i^{--}). Note that such a corner must exist for every block A_i since the corner A_i^{--} coincides with an odd number of other $1/4$ corners (similarly, by Lemma 1). Continue this procedure until we reach block A_n that overlaps block B in at least one of X or Y coordinate, which must happen eventually.

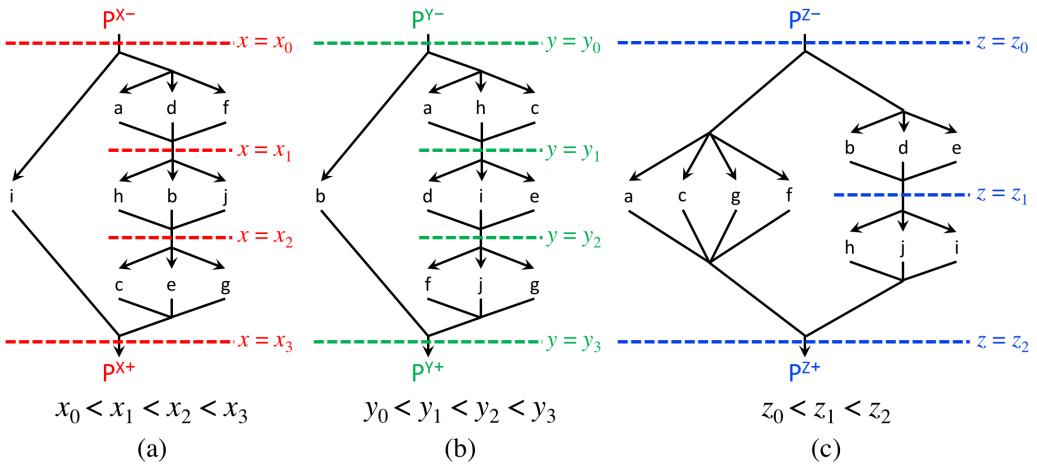
4. If the two blocks A and B do not overlap in any of three coordinates, assume (without loss of generality) that A's coordinates are larger than B's coordinates in each dimension, e.g., $A^{X-} > B^{X+}$, $A^{Y-} > B^{Y+}$, and $A^{Z-} > B^{Z+}$. We produce a chain of blocks $A_0, A_1, A_2, \ldots, A_n$ where $A = A_0$ and block A_{i+1} has an $1/8$ corner that shares the corner with block A_i's $---$ corner (i.e., A_i^{---}). Note that such a corner must exist for every block A_i since the corner A_i^{---} coincides with an odd number of other $1/8$ corners (by Lemma 1). Continue this procedure until we reach block A_n that overlaps block B in at least one of X, Y, or Z coordinate, which must happen eventually.

5. Based on 2, 3, or 4, we obtain $A_0 \geq A_1 \geq A_2 \geq \ldots \geq A_n$ under at least one of dimensions. These inequalities hold for all the partial orders. Since $A = A_0$ and $A_n \geq B$, $A_0 \geq A_n \geq B$. Therefore, we have the relation $A \geq B$ under at least one of the partial order representation.

□

2.4.4 Partial Order Representation to Blocks' Absolute Coordinates

A partial order representation can restore the absolute coordinates of all blocks in the 3-D mosaic floorplan by using the given physical dimensions of blocks. Based on the relative orders of cutting planes per each dimension, the partial order of each dimension provides information on the relative orders of the blocks in the corresponding dimension. For direction d where d is one of X, Y, and Z, (1) we start from the minimum d-directional coordinate of the floorplan space P, i.e., P^{d-}. (2) We iterate through the cutting planes, incrementing a "layer counter" for each cutting plane. (3) We finish the procedure when we reach P^{d+}. (4) Then we obtain the minimum and maximum coordinates of each block in d direction. Examples are shown in Figure 2.17 and Table 2.1 for the 3-D mosaic



$x_0 < x_1 < x_2 < x_3$ (a) $y_0 < y_1 < y_2 < y_3$ (b) $z_0 < z_1 < z_2$ (c)

Figure 2.17: Obtaining X-, Y-, and Z-dimensional relative orders of all blocks in the 3-D floorplan based on the partial order representation in Figure 2.13.

floorplan in Figure 2.7. The relative orders for cutting planes are $x_0 < x_1 < x_2 < x_3$, $y_0 < y_1 < y_2 < y_3$, and $z_0 < z_1 < z_2$ in X-, Y-, and Z-directional partial orders, respectively. Suppose that block B's X-directional physical dimension is $B(X)$. Then we have Equations (2.2) to obtain the absolute coordinates of the entire blocks in X direction. We assume $x_0 = y_0 = z_0 = 0$. Similarly, Y- and Z-directional absolute coordinates can be obtained. In Table 2.1, $Block^{d+}$ and $Block^{d-}$ denote $Block$'s d-directional maximum and minimum coordinates, respectively, where d is X, Y, or Z.

$$\begin{aligned} a(X) = d(X) = f(X) \quad ; \quad x_1 = a(X) \quad ; \\ h(X) = b(X) = j(X) \quad ; \quad x_2 = x_1 + h(X) \quad ; \\ c(X) = e(X) = g(X) \quad ; \quad x_3 = x_2 + c(X) \quad ; \\ x_3 = i(X) \end{aligned} \tag{2.2}$$

Table 2.1: X-, Y-, and Z-directional absolute coordinates of all blocks of the 3-D floorplan in Figure 2.7, based on the relative orders of cutting planes in Figure 2.17. With the given physical dimensions of blocks, we can obtain the entire coordinates as described in Equations (2.2).

Blocks	X Coordinate		Y Coordinate		Z Coordinate	
	$Block^{X-}$	$Block^{X+}$	$Block^{Y-}$	$Block^{Y+}$	$Block^{Z-}$	$Block^{Z+}$
a	x_0	x_1	y_0	y_1	z_0	z_2
b	x_1	x_2	y_0	y_3	z_0	z_1
c	x_2	x_3	y_0	y_1	z_0	z_2
d	x_0	x_1	y_1	y_2	z_0	z_1
e	x_2	x_3	y_1	y_2	z_0	z_1
f	x_0	x_1	y_2	y_3	z_0	z_2
g	x_2	x_3	y_2	y_3	z_0	z_2
h	x_1	x_2	y_0	y_1	z_1	z_2
i	x_0	x_3	y_1	y_2	z_1	z_2
j	x_1	x_2	y_2	y_3	z_1	z_2

2.5 3-D Floorplan Representation Algorithms

In this section, we describe two algorithms: (i) to convert corner links representation to partial order representation (Algorithm 1 in Section 2.5.1), and (ii) to restore the absolute coordinates of the entire blocks in the 3-D floorplan by using the relative orders of cutting planes described by partial order representation (Algorithm 2 in Section 2.5.2).

2.5.1 Corner Links to Partial Order Representation

Algorithm 1 describes the procedure to convert corner links representation to partial order representation. Based on Theorem 2.1, a complete list of the corner links in the 3-D floorplan can be reduced to the three partial orders for each dimension of the

Algorithm 1 Corner Links to Partial Order Representation

Input: Complete set of corner links in the 3-D floorplan, composed of a set of equivalent block corners
Output: Three partial orders for each dimension X, Y, and Z (i.e., three lists of cutting planes that are ordered to each dimension)

/* Note: i-th cornerLink $cL_i \in CornerLinks$ is composed of a set of neighboring corners nC_j, which includes all pairs of $1/8$ corners in the adjacent blocks that intersect at the same coordinate. j-th neighboring corner $nC_j \in cL_i$ is defined as $A^{xyz} = B^{xyz}$ where A and B denote blocks and x, y, z indicate the relative location of the corner as $+$ or $-$. */

Procedure getPartialOrders (CornerLinks)
for each dimension $d \in \{X,Y,Z\}$ **do**
  $sCuttingPlanes(d) \leftarrow \emptyset$;
end for
for each cornerLink $cL_i \in$ CornerLinks **do**
  for each neighboringCorners $nC_j \in cL_i$ **do**
    for each dimension $d \in \{X,Y,Z\}$ of the corner equation of nC_j **do**
      Obtain d-dimensional equivalent corner relation, i.e., cutting plane $cP_i(d)$;
      $sCuttingPlanes(d) \leftarrow sCuttingPlanes(d) \cup$ cuttingPlane $cP_i(d)$;
    end for
  end for
end for
for each $sCuttingPlanes(d)$ where $d \in \{X,Y,Z\}$ **do**
  $partialOrder(d) \leftarrow$ acyclicDirectedGraph($sCuttingPlanes(d)$);
end for

3-D space by using cutting planes. Algorithm 1 shows the conversion procedure from the corner links representation to the partial order representation. From the inputs of the complete set of corner links in 3-D mosaic floorplan, Algorithm 1 develops three partial orders for each dimension X, Y, and Z as its outputs. The resulting partial orders are acyclic directed graphs, i.e., topologically ordered three transitive closure graphs per each dimension.

Note that corner links are composed of a set of neighboring corners, and each corner link has at least two and the even number of $1/8$ corners as shown in Lemma 1, i.e., we have at least a pair of neighboring corners in the corner link. Also all $1/8$ neighboring corners are pair-wise equal by the corner equations. In Algorithm 1, i-th corner link cL_i in the set of the corner links *CornerLinks* is composed of a set of neighboring corners nC_j, which includes all pairs of $1/8$ corners in the adjacent blocks that intersect at the same coordinate. And j-th neighboring corner nC_j is defined with corner equation as $A^{xyz} = B^{xyz}$ where A and B denote blocks determining nC_j and x, y, z indicate the relative location of the corner within the blocks as $+$ or $-$. Through the first FOR-loop, we initialize sets of cutting planes per each dimension, $sCuttingPlanes(d)$, as empty sets. We then proceed the second FOR-loop for each corner link cL_i in a set *CornerLinks* that includes the entire corner links of the given 3-D floorplan. For each corner link cL_i, we traverse every neighboring corner nC_j in cL_i and obtain each dimensional equivalent corner relation from the corner equation of nC_j. This equivalent corner relation determines the cutting plane $cP_{ij}(d)$, which will be added into the set of cutting planes for each dimension d. After traversing all corner links through the second FOR-loop, we have complete sets of cutting planes for each dimension. As every cutting plane defines the relative location of blocks, we can generate each dimensional partial order $partialOrder(d)$ by obtaining the acyclic directed graph of cutting planes in $sCuttingPlanes(d)$.

2.5.2 Partial Order to Absolute Coordinate Representation

Algorithm 2 describes the procedure to restore the absolute coordinates of every block in the 3-D floorplan from the partial orders of blocks for each dimension. As discussed in Section 2.4.3, the partial order representation can restore the absolute coordinates of all blocks in the 3-D floorplan since the partial orders give the topological-

Algorithm 2 Partial Order to Absolute Coordinate

1: **Input**: Three partial orders for each dimension X, Y, and Z
2: **Output**: Absolute coordinates of $---$ and $+++$ corners of each block

3: **Procedure getCoordinateForAllBlocks (partialOrders)**
4: **for** each $partialOrder(d) \in$ partialOrders **do**
5: /* i.e., for each dimension d where $d \in \{X,Y,Z\}$ */
6: initialPlane $iP \leftarrow$ floorplanSpace.$d-$;
7: $iP.d-$.coordinate $\leftarrow 0$;
8: $sBlock \leftarrow \{A_i \mid$ blocks that are connected from iP plane in $partialOrder(d)\}$;
9: **for** each block $A_i \in sBlock$ **do**
10: /* Block A_i is i-th block of $sBlock$ */
11: $A_i.d-$.coordinate $\leftarrow iP.d-$.coordinate;
12: **end for**
13: **while** $sBlock \neq \emptyset$ **do**
14: getPlusCoordinate ($sBlock$);
15: updateSetBlock (sBlock);
16: **end while**
17: **end for**

18: **Procedure getPlusCoordinate (sBlock)**
19: **for** each block $A_i \in sBlock$ **do**
20: $A_i.d+$.coordinate $\leftarrow A_i.d-$.coordinate + $A_i.d$.dimension;
21: **end for**

22: **Procedure updateSetBlock (sBlock)**
23: **for** each block $A_i \in sBlock$ **do**
24: **if** ($A_i.d+$ plane touches floorplanSpace.$d+$) **then**
25: $sBlock \leftarrow sBlock \setminus A_i$;
26: **continue**;
27: **end if**
28: $sNewBlock \leftarrow \{B_j \mid$ blocks that are faced with $A_i.d+$ plane in $partialOrder(d)\}$;
29: **for** each block $B_j \in sNewBlock$ **do**
30: $B_j.d-$.coordinate $\leftarrow A_i.d+$.coordinate;
31: **end for**
32: $sBlock \leftarrow sBlock \cup sNewBlock$;
33: $sBlock \leftarrow sBlock \setminus A_i$;
34: **end for**

relative orders of all blocks for each dimension. For each dimensional partial order $partialOrder(d)$, Algorithm 2 starts with the initialPlane iP as the minimum plane of the floorplan space floorplanSpace.$d-$, i.e., the root node of each dimensional partial or-

der. We define the absolute coordinates of these root nodes as 0. In Algorithm 2, *sBlock* is a set of blocks to be considered in each WHILE-loop iteration, and is initialized as a set of blocks that are connected from *iP* plane in $partialOrder(d)$. For each block $A_i \in sBlock$, we determine the minimum coordinate of block A_i (i.e., $A_i.d-$.coordinate) as the minimum coordinate of the 3-D floorplan space (i.e., $iP.d-$.coordinate). We execute WHILE-loop until *sBlock* becomes empty set. WHILE-loop has two subprocedures. Procedure **getPlusCoordinate (sBlock)** obtains the maximum coordinates of each block A_i in *sBlock*, which adds A_i's d dimension to block A_i's minimum coordinate. Then Procedure **updateSetBlock (sBlock)** is performed for each block $A_i \in sBlock$. The procedure first removes block A_i from *sBlock* if block A_i touches the maximum coordinate of the given 3-D floorplan. Otherwise, the procedure separately creates a set of blocks *sNewBlock* containing all blocks faced with the maximum coordinate of block A_i. For each block $B_j \in sNewBlock$, we define the minimum coordinates of block B_j as the maximum coordinate of block A_i. Then the procedure adds all blocks in *sNewBlock* into *sBlock*, and removes block A_i from *sBlock*. We obtain the absolute coordinates of all blocks when we terminate WHILE-loop.

2.6 Conclusion

In this chapter, we have presented our new 3-D floorplan representation, *corner links representation*. We define corner links representation as the spatial relations that describe all corner relations of the entire blocks in the 3-D mosaic floorplan, where a corner link is a set of $1/8$ neighboring corners (i.e., vertices) that belong to the adjacent blocks next to each other in X, Y, Z, or diagonal directions. We have analyzed corner links' key properties with lemmas, theorems, and their proofs along with that of the existing *partial order representation*, i.e., three transitive closure graphs for each dimension. (1) Corner links representation can be reduced to the partial order representation. (2) A non-degenerate 3-D mosaic floorplan can be equivalently expressed by the corresponding four trees representation. (3) A partial order representation with three transitive closure graphs captures all cutting planes in the 3-D mosaic floorplan, in order of their respective dimensions. (4) The 3-D floorplan is a valid floorplan if the partial order representation describes relations between all pairs of blocks in the 3-D mosaic floorplan. (5) A partial

order representation can restore the absolute coordinates of all blocks in the 3-D mosaic floorplan by using the given physical dimensions of blocks. We leave to our future work a couple of directions: (i) implementation and verification of the real 3-D floorplan representation for 3-D IC design; (ii) more encoding schemes and further reduce the total number of combinations while preserving the completeness of the encoding, and (iii) extension of our representation methodology to four (4-D) or even higher dimensions for the mapping of dynamic re-programmable 3-D devices, the thermal management minimizing the peak temperature, etc.

2.7 Acknowledgments

Chapter 2 contains a reprint of Fang Qiao, Ilgweon Kang, Daniel Kane, Evangeline Fung Yu Young, Chung-Kuan Cheng and Ronald Graham, "3D Floorplan Representations: Corner Links and Partial Order", *Proc. IEEE International Conference of 3D System Integration*, 2016. Chapter 2 also contains the draft submitted to *ACM Transactions on Design Automation of Electronic Systems*, Ilgweon Kang, Fang Qiao, Dongwon Park, Daniel Kane, Evangeline Fung Yu Young, Chung-Kuan Cheng and Ronald Graham, "3-D Floorplan Representations by Using Corner Links and Partial Order", 2018. The dissertation author is a main contributor of the paper and the primary author of the submitted draft.

I would like to thank my coauthors Fang Qiao, Dongwon Park, Daniel Kane, Evangeline Fung Yu Young, Chung-Kuan Cheng and Ronald Graham.

Chapter 3

Advancing Solution Quality and Routability Validation in Global Placement

The Nesterov's method approach to analytic placement [79] [80] [81] has recently demonstrated strong solution quality and scalability. We dissect the previous implementation strategy of [81] and show that solution quality can be significantly improved using two levers: *constraint-oriented local smoothing*, and *dynamic step size adaptation*. We propose a new density function that comprehends local overflow of area resources; this enables a constraint-oriented local smoothing at per-bin granularity. Our improved dynamic step size adaptation automatically determines step size and effectively allocates optimization effort to significantly improve solution quality without undue runtime impact. Our resulting global placement tool, RePlAce, achieves an average of 2.00% HPWL reduction over all best known ISPD-2005 and ISPD-2006 benchmark results, and an average of 2.73% over all best known MMS benchmark results, without any benchmark-specific code or tuning. We further extend our global placer to address routability, and achieve on average 8.50% to 9.59% scaled HPWL reduction over previous leading academic placers for the DAC-2012 and ICCAD-2012 benchmark suites. To our knowledge, RePlAce is the first work to achieve superior solution quality across all the ISPD-2005, ISPD-2006, MMS, DAC-2012 and ICCAD-2012 benchmark suites with a single placement engine.

3.1 Introduction

Placement is a fundamental, critical step in the physical design (PD) of integrated circuits (ICs) [60]. Placement solution quality directly impacts overall design quality of results with respect to timing closure, die utilization, routability, and design turnaround time; these in turn affect the classic metrics of operating frequency, yield, power consumption and cost. Despite significant improvement in placement algorithms over the past decades [89], efficient and effective placement remains a challenging issue [5].

Among all academic placers, recent electrostatics-based placement (ePlace) implementations [79] [80] [81] achieve benchmark solutions that rank among the best known in terms of half-perimeter wirelength (HPWL). ePlace is a flat, nonlinear analytical global placement engine with electrostatics-based global-smooth density cost function and Nesterov's method nonlinear optimizer. The density cost function enables effective movement of standard cells and macros over fixed instances, blockages and large macros. The density cost is solved numerically by a fast Fourier transform (FFT) [113] with high accuracy and $O(n \log n)$ complexity. Nesterov's method provides accelerated convergence, with steplength dynamically predicted via Lipschitz constant. A backtracking method effectively prevents steplength overestimation. ePlace is capable of standard-cell placement [80], mixed-size placement [81], and 3D-IC mixed-size placement [82]. Instance size differences between standard cells and macros are addressed by an approximated nonlinear preconditioner. Yet despite these and other previous efforts, our work demonstrates the availability of significant further improvement over *best known* HPWL results for standard academic benchmarks. Furthermore, ePlace [79] [80] [81] cannot produce routable placements, e.g., for *SUPERBLUE12* [121] ePlace routing hotspots demand 211.73% of the routing supply (contrast this with RePlAce's RC value of 102.43% in Table 3.7). In RePlAce, we add optimization of routability in global routing to the Nesterov's approach, achieving substantial *scaled HPWL* improvements over previous leading academic placers for the DAC-2012 [121] and ICCAD-2012 [122] benchmark suites.

3.1.1 Density function and density penalty factor

Conventional global placement methodology seeks to minimize wirelength subject to density constraints which mitigate instance overlaps. The density constraints can be transformed to yield an unconstrained objective with a density penalty factor, as shown in Equation (3.1). Previous nonlinear placers [19] [63] [80] apply the density penalty factor globally across the entire placement region, with the penalty factor increased proportionally [63] [80], or at a constant rate [19], until the end of global placement. Such approaches suffer from the "global" nature of their iterations, which can overlook the fine-grain spatial and temporal behavior of the placement procedure. In other words, a globally-applied penalty factor can be insensitive to density variations across the placement region, and a fixed schedule for growth of the density penalty factor will not discern between early and late stages of global placement. This can lead to unnecessary suboptimality of solutions.

3.1.2 Routability-driven placement

Routability is a fundamental requirement of real-world global placement [4] [121] [122], as the placement process must provide a routable placement solution to the router. It is well-understood that the standard minimum total HPWL placement objective at some point becomes detrimental to routability. Previous works achieve improved routability via (i) congestion estimation, and (ii) congestion mitigation. Several works [12] [126] [140] are based on placement properties (e.g., Rent's parameter, pin density, net overlapping, etc.), without considering actual routing. Probabilistic estimations assume a uniform wire density model [114], or pattern routing considering wire bends and vias [131]. More recent and effective constructive estimations used in recent routability-driven placers [43] [66] [75] are based on global routers [52] [77] [100]. To mitigate congestion, the works of [47] [56] [114] formulate a routability-driven objective function with multiple Lagrangian multipliers. The works of [42] [43] [66] [75] implement cell inflation with local refinement, or a rough legalizer during global placement, to spread overlapped instances. The DAC-2012 [121] and ICCAD-2012 [122] routability-driven global placement benchmark suites are the most recent academic evaluation frameworks that ad-

dress the routability issue at the global routing stage, and are widely used to validate the performance of academic placers. Among all published results for these two benchmark suites, [24] [43] [47] [75] show leading-edge solution qualities in terms of scaled HPWL, considering routing congestion (RC) as a penalty factor to HPWL as defined in [121] [122]. A separate body of work (e.g., [27] [54]) addresses the routing-driven ISPD-2014 [146] and ISPD-2015 [11] benchmarks. However, quality of results heavily depends on detailed placers that are sensitive to details of technology and library cells, which is beyond scope for a global placement framework such as ours.

In our work, we achieve global placement solution quality well beyond published (best known) results for the ISPD-2005, ISPD-2006 and MMS benchmark suites via a new constraint-oriented local-density function (RePlAce-ld), and an improved dynamic step size adaptation (RePlAce-ds, RePlAce-ldds). With extensions to support routability as assessed in global routing (RePlAce-r), we achieve superior solution quality on the DAC-2012 and ICCAD-2012 benchmark suites. Our contributions are as follows.

- We propose a new constraint-oriented local-density function for mixed-size placement that incorporates (i) a constraint-oriented local-density penalty factor for each bin (i.e., local Lagrangian multiplier for each bin), and (ii) a constraint-oriented local-density cost coefficient for each instance. Combining the previous global density function [81] with a new local density function that comprehends local *density overflow* per bin[2], we obtain a global placement with *constraint-oriented local smoothing* that achieves improved solution quality.

- We propose a methodology for density-penalty adaptation via an improved *dynamic step size adaptation* that automatically adjusts the density penalty factor based on the *HPWL curve* (i.e., trajectory of HPWL cost versus iteration count) observed in a *trial placement procedure*.[3] Our improved dynamic step size adaptation applies more fine-grained control at transition points on the HPWL curve. Compared to a constant small step size, we obtain better solution quality while saving runtime.

[2]Density overflow for a given placement bin is defined as the total area of instances inside the placement bin, minus the placement bin area.

[3]As we describe in detail below, a trial placement procedure is performed initially to capture *transition points* on the HPWL curve; these transition points inform the step size adaptation.

- We validate RePlAce by HPWL comparison to all best known ISPD and MMS benchmark results. Without any testcase-specific tuning, we achieve an average HPWL reduction of 2.00% over the best known ISPD benchmark results, and of 2.73% over the best known MMS benchmark results.

- We propose a layer-aware cell inflation technique, considering per-layer pin blockages, and integrate the official global router *NCTU-GR* [161] of the DAC-2012 and ICCAD-2012 benchmark suites for congestion estimation. We develop a simple but effective superlinear cell inflation technique to mitigate global routing congestion during global placement. Following the strategy of recent leading works [42] [43], we further include a post-placement optimization by [76]. By integrating all our innovations to improve routability, our placer delivers solution quality in terms of scaled HPWL that substantially improves over previous leading academic placers for the DAC-2012 and ICCAD-2012 benchmark suites, achieving on average 8.50% to 9.59% scaled HPWL reduction over previous placers.[4]

The remainder of this chapter is organized as follows. Section 3.2 briefly states the fundamental placement problem formulation. Section 3.3 introduces our constraint-oriented local-density function with local Lagrangian multiplier, achieving local smoothing. Section 3.4 describes our improved dynamic step size adaptation. Section 3.5 describes our methodology to improve routability, with congestion estimation by a global router and a cell inflation technique. Section 3.6 presents our experimental setups and results. Section 3.7 concludes this chapter.

[4]We use these benchmarks instead of the ISPD-2014 and ISPD-2015 benchmarks since our focus is on mitigating congestion reported by the global router, along the lines of well-addressed, industry-formulated routability-driven global placement contests [121] [122]. From a practical IC implementation flow (i.e., turnaround time) standpoint, global placement-based mitigation of (global) routing congestion remains crucially important. In Section 3.6.3 below, a brief comparison between RePlAce and a leading-edge commercial placer suggests that remaining "gaps" between academic research and industry practice are potentially tractable in today's university research context.

3.2 Placement Overview

Placement seeks to determine the location of instances (e.g., standard cells and macros) while addressing optimization objectives such as HPWL, routed wirelength, timing, power, routability, etc. A placement solution is represented as

$$\mathbf{v} = (\mathbf{x},\ \mathbf{y})^T = (x_1,\ x_2,\ \cdots,\ x_n;\ y_1,\ y_2,\ \cdots,\ y_n)^T,$$

where $(x_i,\ y_i)$ is the physical location (of the origin, with orientation) of the i^{th} instance. A legal placement solution requires that (i) every instance is placed in the placement region within predefined rows; and (ii) instances do not overlap. We follow the basic notations in [81] and formulate the placement objective function as shown in Equation (3.1).

$$\min_{\mathbf{v}} f(\mathbf{v}) = W(\mathbf{v}) + \lambda D(\mathbf{v}) \tag{3.1}$$

The wirelength objective $W(\mathbf{v})$ is the HPWL of the design modeled with a weighted-average (WA) smoothing technique [46], while the density cost function $D(\mathbf{v})$ addresses instance overlap via an electrostatic analogy [81]. During nonlinear optimization, a density penalty factor (i.e., global Lagrangian multiplier) λ is gradually increased to reduce overlap, at the cost of increased wirelength.

3.3 Constraint-Oriented Local-Density Function

We now describe our improvement of the previous electrostatics-based density formulation in [79] [80] [81].

3.3.1 Necessity of Local Density Function

Scaling the density penalty factor in the placement objective function is critical since this directly impacts the placement solution quality. Like many other analytical placers [19] [45], the previous ePlace implementations of [79] [80] [81] use the density penalty factor λ, applied equally to every placement grid or bin in each iteration, to balance wirelength and density costs. However, in our work, we make the motivating observation that

globally applying the density penalty factor (i.e., applying global Lagrangian multiplier) sacrifices wirelength in less-overlapped bins to resolve more-overlapped bins, resulting in unnecessary wirelength increase.

With this in mind, we propose a new *local-density* function that comprehends locally overflowed (with respect to area resources) bins; this enables *constraint-oriented local smoothing* at a per-bin granularity. To help effective removal of overlaps among placement instances, our local-density function (per each bin) provides more repulsive forces[5] for overflowed bins, on the basis of the global-smooth density distribution obtained by global-density function. This is beneficial since the local-density function on top of the global-density function effectively helps us restrain suboptimal wirelength increase caused by global-density penalty factor λ. In this section, we use the term *global* when we refer to a density penalty that is applied equally throughout the layout, and *local* when we refer to penalty factors or coefficients that are separately defined and used at a per-bin granularity. We denote our new local-density function as $D^{local}(\mathbf{v})$, and the previous global-density function of [79] [80] [81] as $D^{global}(\mathbf{v})$.

With respect to Equation (3.1), we begin with $D(\mathbf{v}) = D^{global}(\mathbf{v})$, and then add a further term $D^{local}(\mathbf{v})$ to introduce the local smoothing to our placement engine. The local density function incorporates two innovations. First, we formulate a *constraint-oriented local-density penalty factor* ν_j (i.e., local Lagrangian multiplier) per bin to spread cells in highly overflowed regions by increased repulsive forces (Section 3.3.2). Second, we apply a *constraint-oriented local-density coefficient* Δ_i per instance i to the local-density function (Section 3.3.3). Δ_i helps the instance i maintain a certain amount of repulsive force induced from overflowed bins, even as the instance i escapes from (i.e., is no longer contained in) those overflowed bins.

3.3.2 Constraint-Oriented Local-Density Penalty for Each Bin b_j

To enable the constraint-oriented local smoothing, we introduce a local-density penalty factor ν_j (i.e., local Lagrangian multiplier) per each bin b_j based on demands for

[5] We note that in our electrostatic analogy, the gradient of the cost function is the repulsive force from electric charges. We use "force" to refer to this repulsive force due to electric charges.

area resources. We formulate the local-density penalty factor ν_j as

$$\nu_j = e^{\alpha \cdot (BinDemand_j - BinCapacity_j)}. \tag{3.2}$$

In Equation (3.2), $BinCapacity_j$ and $BinDemand_j$ respectively denote the area of bin b_j and the total area of cells intersecting b_j. We define b_j's *overflow* as ($BinDemand_j - BinCapacity_j$). The bin b_j is *overflowed* if $BinDemand_j - BinCapacity_j > 0$. α is a coefficient to weight the local-density cost function as detailed in Equations (3.5) and (3.6) below; α starts at a very small value, e.g., $1e$-12 (empirically determined), and gradually increases through the Nesterov's optimization. When a bin b_j is overflowed, ν_j has exponentially larger value, generating larger repulsive force. Thus, cells in b_j experience larger force to be spread toward not-overflowed bins. When b_j is not overflowed, cells in b_j experience small force. The local-density penalty factor ν_j is especially beneficial early in the placement procedure since it guides cells to quickly find their directions of movement.[6] Consequently, the local-density penalty factor ν_j of each bin b_j helps our RePlAce to spread cells quickly for those cells on highly over-demanded bins.

Figure 3.1 illustrates the benefit of the constraint-oriented local-density penalty factor ν_j. Each i represents a placement instance (cell) belonging to the same (5-pin) net, and red arrows indicate the repulsive force to spread cells. The faint outlines show the previous locations of cells. In Figure 3.1(a), the global λ is applied equally to all cells in the layout, helping to remove overlap between i_5 and $Macro_1$. In Figure 3.1(b), the local smoothing is applied with local density-penalty factor ν_j per bin, so that HPWL increases (from the dotted to the solid blue rectangle) less than in (a). To remove the overlap between i_5 and $Macro_1$, in (a) the global density function $D^{global}(\mathbf{v})$ is applied to all cells, scaled uniformly by the global density penalty factor λ, even though most of the affected cells are not in the overflowed bins. The repulsive forces induced by $D^{global}(\mathbf{v})$ and λ cause a large HPWL increment, depicted by the transition between dotted and solid blue rectangles. By contrast, in (b) the constraint-oriented local-density penalty factor ν_j separately scales the magnitude of the repulsive force per each bin based on that bin's overflow, i.e., bins b_2 and b_4 have relatively larger ν_j to resolve overlaps. In this way,

[6] Initial placement typically seeks only to minimize wirelength, which results in a number of highly overflowed bins.

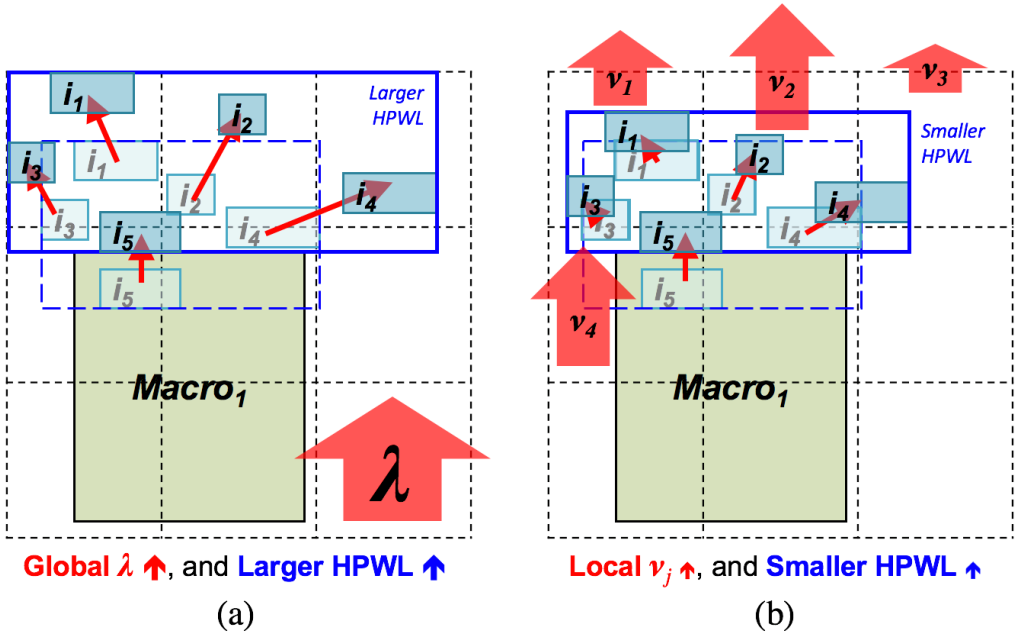


Figure 3.1: Density forces with (a) global density-penalty factor λ, and (b) constraint-oriented local-density penalty factor ν_j per each bin. (Bin boundaries are indicated by black dotted lines. Standard-cell instances are labeled $i_1, \cdots, i_5$.)

the local density suppresses the increase of the global Lagrangian multiplier, and bins b_1 and b_3 could experience smaller density force than in Figure 3.1(a), resulting in a smaller HPWL increment.

3.3.3 Local-Density Cost Coefficient Δ_i per Each Cell i

With constraint-oriented local smoothing, per Equation (3.2), cell i from an overflowed bin immediately loses much of its repulsive force component after it escapes the overflowed bin. Without sufficient repulsive force induced from the global density function, the movement of cell i slows down if the adjacent bin to which it moves is not overflowed. Furthermore, the cell i can return to its previous, (formerly) overflowed bin as a consequence of wirelength-induced attractive force. In such a scenario, instances (cells) are not effectively spread to resolve the cell overlapping. To effectively achieve local smoothing via a local density penalty factor without globally sacrificing HPWL, we propose a new mechanism to maintain the repulsive forces generated by the constraint-oriented local-density penalty factor ν_j of the overflowed bin b_j.

Equation (3.3) gives the *constraint-oriented* local-density cost coefficient, Δ_i, per each cell i (c_i, $c_i \in b_j$). We use the convention $c_i \in b_j$ to indicate that cell i intersects bin b_j. Δ_i is used to multiplicatively scale the local density function $D^{local}(\mathbf{v})$.

$$\Delta_i^{iter+1} = \Delta_i^{iter} + \beta \cdot \frac{\max(Overflow_j, 0)}{\sum_i A_i} \tag{3.3}$$

Here, A_i is the area of cell i (c_i) and $iter$ is the index of the iteration. We initialize $\Delta_i = 0$. With respect to the current overflow of bin b_j, Δ_i accumulates positive overflow normalized by the total cell area in the design (i.e., $\sum_i A_i$). β is a coefficient to balance between global-density and local-density induced forces. β initially takes on a very small value, e.g., $1e$-13 (empirically determined), and gradually increases through the Nesterov's optimization.[7] Multiplicatively scaling the local-density function by Δ_i prevents cell i from losing its local-density penalty factor induced from the repulsive force, even as cell i moves out of the overflowed bin b_j. (See Equation (3.9) and Algorithm 3)

Figure 3.2 illustrates the advantage of multiplying by the constraint-oriented local-density cost coefficient Δ_i. Bin b_4 is overflowed, while the other bins are not overflowed. Red arrows depict the repulsive force component induced from the local-density penalty factor ν_j. In Figure 3.2(a), without Δ_i, the magnitude of force rapidly decreases after a cell i escapes from the overflowed bin b_4, resulting in slower movements. In Figure 3.2(b), the local-density cost coefficient Δ_i compensates the loss of the repulsive force component induced from the overflowed bins. Blue arrows in Figure 3.2(b) indicate the repulsive force components induced from the local-density cost coefficient Δ_i in each iteration.

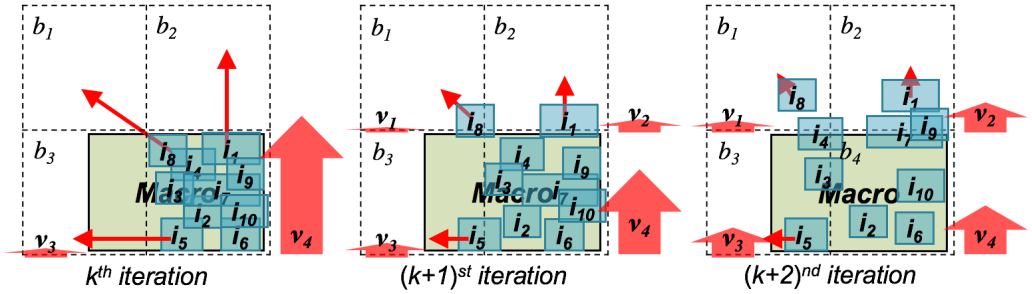
3.3.4 Formulation: Local Density Function and Gradient

Table 3.1 summarizes our notations.[8] Equation (3.4) shows the *global density function* introduced in [81],

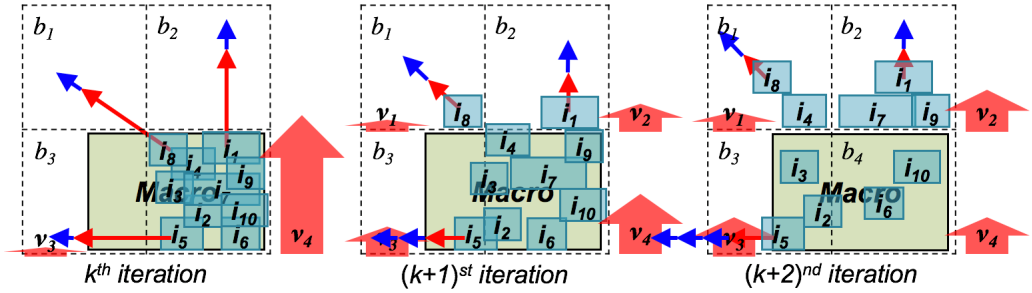
$$D^{global}(\mathbf{v}) = \sum_i D_i^{global}(\mathbf{v}) = \sum_i q_i \phi_i(\mathbf{v}), \tag{3.4}$$

[7] $\beta^{iter+1} = cof \times \beta^{iter}$ where cof is the step size defined in Section 3.4.1.

[8] ϕ_j and $\mathbf{E}_j$ are the electric potential and field at bin j, respectively. They are calculated using the existing charge density (from current placement of all instances) by FFT. Due to the discrete nature of FFT, the electric potential and field has a discrete value at per-bin granularity. Thus, $\mathbf{E}_i$ for instance i is $\mathbf{E}_j$ where instance i is at bin j.



(a) Local smoothing without local-density cost coefficient Δ_i.



(b) Local smoothing with local-density cost coefficient Δ_i.

Figure 3.2: Local smoothing methods (a) without and (b) with local-density cost coefficient Δ_i. Figures are ordered from left to right by iteration indices. Figure (b) shows the effect of a larger force to spread cells from the overflowed bin b_4.

where q_i is the electric charge and ϕ_i is the potential of cell i. $D^{global}(\mathbf{v})$ is equal to $D(\mathbf{v})$ in Equation (3.1).

To achieve local smoothing, we add the *local-density function* $D^{local}(\mathbf{v})$ as a further term to our placement objective function. Using the constraint-oriented local-density penalty factor ν_j, we formulate the new local-density function $D^{local}(\mathbf{v})$ as

$$D^{local}(\mathbf{v}) = \sum_{b_j \in B} \nu_j D_j^{local}(\mathbf{v}) = \sum_{b_j \in B} \nu_j \Big(\sum_{c_i \in b_j} A_{ij} q_i \phi_j \Big), \tag{3.5}$$

where A_{ij} is the overlapped area between cell i and bin b_j, q_i is again the electric charge of cell i, and ϕ_j is the local electric potential of bin b_j.

Our new placement objective function $f(\mathbf{v})$ incorporates the global density function $D^{global}(\mathbf{v})$ of Equation (3.4) and the local density function $D^{local}(\mathbf{v})$:

$$\min_{\mathbf{v}} f(\mathbf{v}) = W(\mathbf{v}) + \lambda D^{global}(\mathbf{v}) + D^{local}(\mathbf{v}) \tag{3.6}$$

Table 3.1: Notations for local-density function.

Term	Description
i	Index of the i^{th} placement instance (cell), $i = 1, ..., n$
c_i	i^{th} placement instance (cell), $i = 1, ..., n$
b_j	j^{th} bin in the placement region
B	the set of placement bins
q_i	Electric charge of the i^{th} placement instance
ϕ_j	Electric potential at bin j
ϕ_i	Electric potential at the location of i^{th} placement instance
$\mathbf{E}$	Gradient of the potential ϕ, i.e., electric field
A_{ij}	Overlap area between i^{th} placement instance and bin b_j
λ	Lagrangian multiplier for global density cost function
ν_j	Constraint-oriented local-density penalty factor of bin b_j
Δ_i	Local-density cost coefficient of the i^{th} placement instance

By differentiating Equation (3.6), we obtain the force to spread instances (cells and macros). Equation (3.7) gives the gradient of the global density function $D^{global}(\mathbf{v})$ described in [81]. Equation (3.8) gives the gradient of the local density function $D^{local}(\mathbf{v})$

$$\frac{\partial D^{global}(\mathbf{v})}{\partial x_i} = q_i \frac{\partial \phi_i}{\partial x_i} = q_i \mathbf{E}_i(\mathbf{v}), \tag{3.7}$$

$$\begin{aligned} \frac{\partial D^{local}(\mathbf{v})}{\partial x_i} &= \sum_{b_j \in B} \alpha \frac{\partial BinDemand_j}{\partial x_i} \nu_j \sum_{k \neq i, k \in b_j} A_{kj} q_k \phi_j \\ &+ \sum_{b_j \in B} \nu_j \left(\frac{\partial A_{ij}}{\partial x_i} q_i \phi_j + \sum_{k \neq i, k \in b_j} A_{kj} q_k \mathbf{E}_j \right), \end{aligned} \tag{3.8}$$

where $\mathbf{E}$ is the electric field. $Cost_i$ in Equation (3.9) determines the cell i's movement, comprehending both the global density distribution and the local overflowed region.

$$Cost_i = \frac{\partial W(\mathbf{v})}{\partial x_i} + \lambda \frac{\partial D^{global}(\mathbf{v})}{\partial x_i} + \Delta_i \frac{\partial D^{local}(\mathbf{v})}{\partial x_i} \tag{3.9}$$

Algorithm 3 Local-Density Cost Function

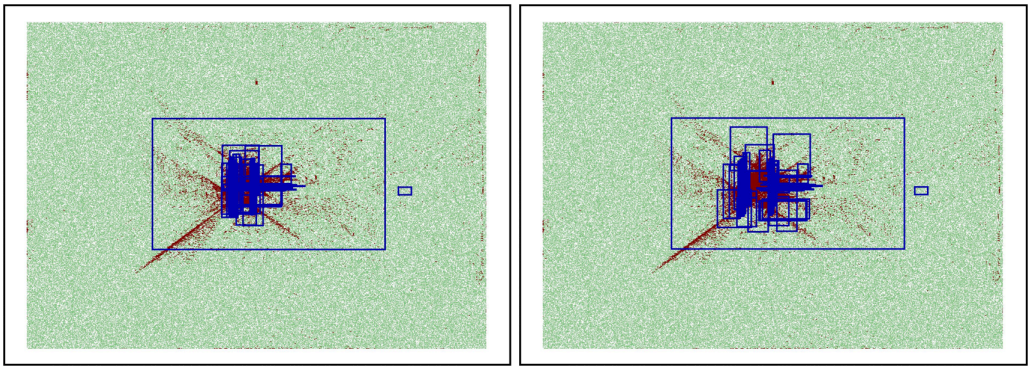
1: **Procedure** $LocalDensityCostFunction(c_i)$
2: Initialize $\Delta_i \leftarrow 0$;
3: **for all** b_j ($b_j \cap c_i \neq 0$) **do**
4: $\quad overflow_j \leftarrow BinDemand_j - BinCapacity_j$;
5: $\quad$ **if** $overflow_j > 0$ **then**
6: $\quad\quad \Delta_i$ += $\beta \times \frac{overflow_j}{total\ cell\ area}$;
7: $\quad$ **end if**
8: $\quad LDCost_i$ += $\Delta_i \times Gradient_j(c_i \in b_j)$;
9: **end for**
10: **return** $Gradient_i$;
11: **Procedure** $Gradient_j(c_i \in b_j)$
12: calculate $Gradient_j$ [81];
13: **return** $Gradient_j$;

Algorithm 3 describes the procedure to obtain the local-density cost, $LDCost_i$. Line 2 initializes $\Delta_i = 0$. We store the electric potential information at a per-bin granularity, so we first calculate $Gradient_j$ to compute $LDCost_i$ (Line 8 and Lines 11-13). Based on cell/charge/energy distribution of the previous iteration, we calculate $Gradient_j$ by using FFT [81] (Lines 11-13). In Lines 5-7, we add the normalized $overflow_j$ of the bin b_j ($c_i \in b_j$) to Δ_i when the bin b_j is overflowed. In Line 8, we obtain $LDCost_i$ after multiplying by Δ_i.

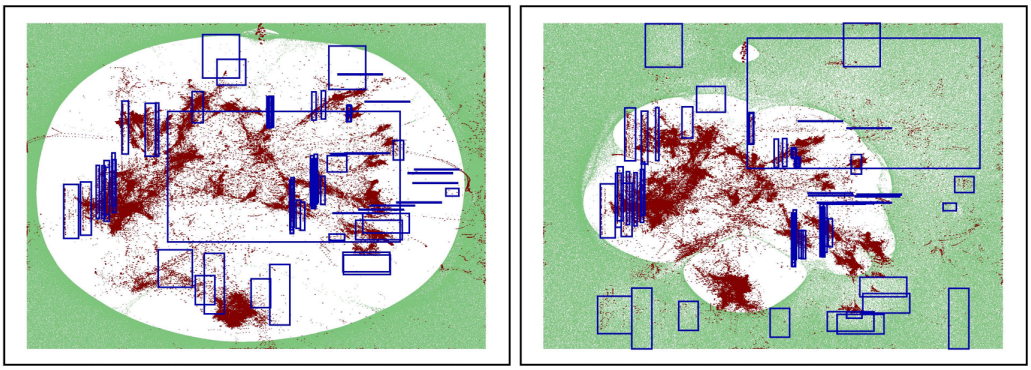
3.3.5 Additional Details and Illustration

Figures 3.3(a)-(c) and Figures 3.4(d)-(e) contrast the mixed-size global placement for *NEWBLUE1* [138], by ePlace-MS [81] (left-side images) and the constraint-oriented local-density function equipped RePlAce-ld (right-side images).[9] Red and green dots represent standard cells and filler cells, respectively. Blue rectangles are movable macros. RePlAce-ld encourages faster cell spreading in overflowed regions by virtue of the repulsive force induced from the local density function. Figures 3.3(a)-(c) show the mixed-size global placement stage (mGP) with movable standard cells and macros. The figures show how ePlace-MS [81] fails to move the largest macro due to lack of force, while RePlAce-ld moves the largest macro toward the boundary of the layout. After mGP, we execute

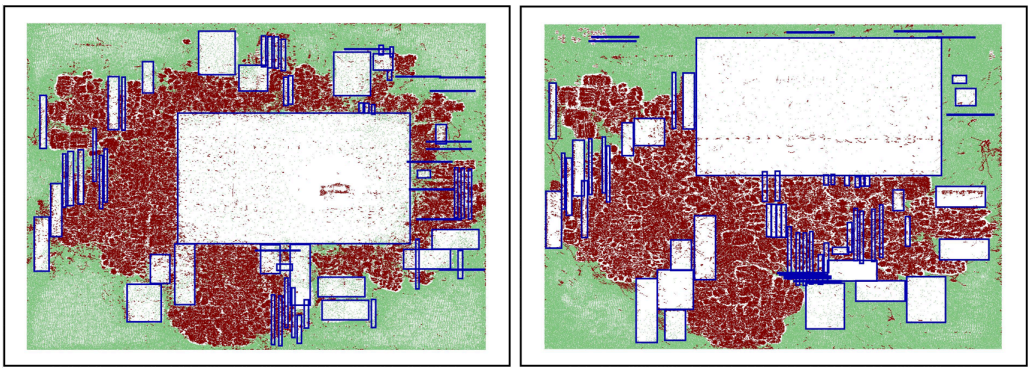
[9] We follow the same mixed-size placement procedure and naming convention as detailed in [81].



(a) Iteration: 250 (mGP), HPWL= 2.07×10^7 (LHS), 2.14×10^7 (RHS).

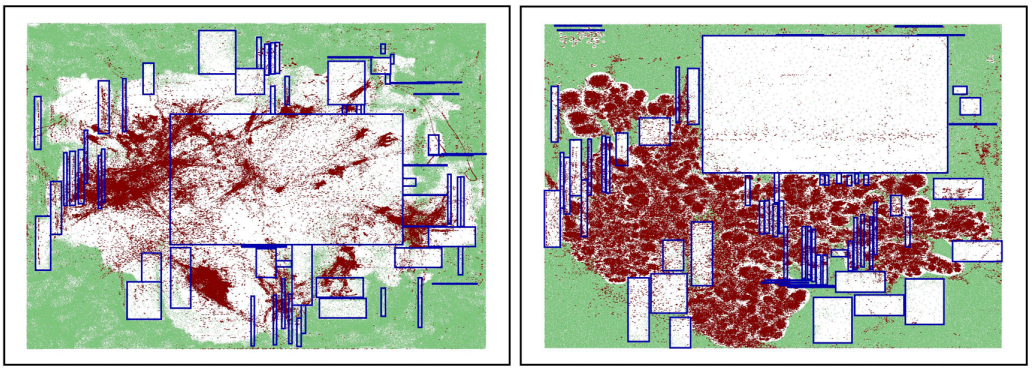


(b) Iteration: 400 (mGP), HPWL= 4.51×10^7 (LHS), 4.32×10^7 (RHS).

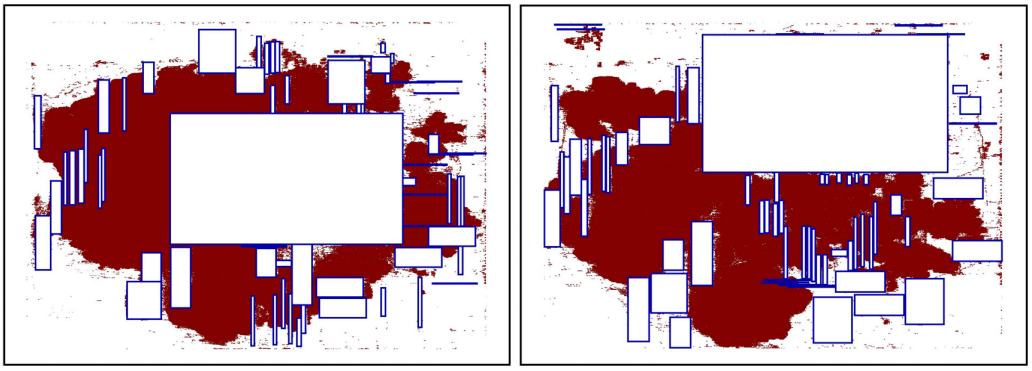


(c) Iteration: 600 (mGP), HPWL= 6.55×10^7 (LHS), 5.32×10^7 (RHS).

Figure 3.3: Placement of *NEWBLUE1* [138]: left hand side (LHS) images are from ePlace-MS [81], and right hand side (RHS) images are from RePlAce-ld. The target density is set to 100%.



(d) Iteration: 30 (cGP), HPWL= 5.18×10^7 (LHS), 5.96×10^7 (RHS).



(e) Final legalized layout, HPWL= 6.39×10^7 (LHS), 5.60×10^7 (RHS).

Figure 3.4: Placement of *NEWBLUE1* [138]: left hand side (LHS) images are from ePlace-MS [81], and right hand side (RHS) images are from RePlAce-ld. The target density is set to 100%. Continued.

a simulated annealing-based macro legalization stage (mLG) to fix all macro locations. Then, a standard-cell only global placement stage (cGP, inheriting instance locations from mLG with macros fixed) is called to recover solution quality lost during macro legalization. Figure 3.4(d) shows that moving the largest macro to the boundary of the layout helps to improve the solution quality by providing more space in the center of the layout region for the standard-cell placement. Moreover, Figure 3.3(b) (left) shows that ePlace-MS applies only one large center-oriented force to filler cells. By contrast, RePlAce-ld applies multiple overflowed-region-induced force components to filler cells, as a result of the local-density function. Figure 3.4(e) shows the final legalized layouts. HPWL values for ePlace-MS and RePlAce-ld are 6.39×10^7 and 5.60×10^7, respectively.

3.4 Improved Dynamic Step Size Adaptation

In this section, we describe our *improved dynamic step size adaptation*. How to decide the step size is a well-studied topic in the global placement literature [18] [19]. In general, constant large step size leads to faster convergence, but can sacrifice solution quality (Figure 3.5(a)). Constant small step size generates an accurate solution according to the given formulation, but with large runtime (Figure 3.5(b)). Neither strategy accounts for the shape characteristics (e.g., instantaneous slope at different iterations) of the *HPWL curve*[10] shown in the figure. We start and end with small step size since we have observed that the step sizes in early and last iterations significantly impact on the placement solution quality. Also we observe that *transition points*[11] on HPWL curve are crucial to obtain better solution quality. Thus our improved dynamic step size adaptation strategy invests more iterations near those transition points on HPWL curve by shrinking the range of step sizes such that the maximum step size is suppressed down to hundreds times smaller compared to our default procedure. Figure 3.5(c) conceptually illustrates our improved dynamic step size adaptation that comprehends the characteristics of the HPWL curve. To effectively

[10] We define the *HPWL curve* as the plot of HPWL values versus iterations in the global placement procedure.

[11] In our improved dynamic step size adaptation, we define and use two classes of transition points based on instantaneous rate of slope change: 1st-order transition point TP^1 and 2nd-order transition point TP^2. The transition points are described in Section 3.4.2.

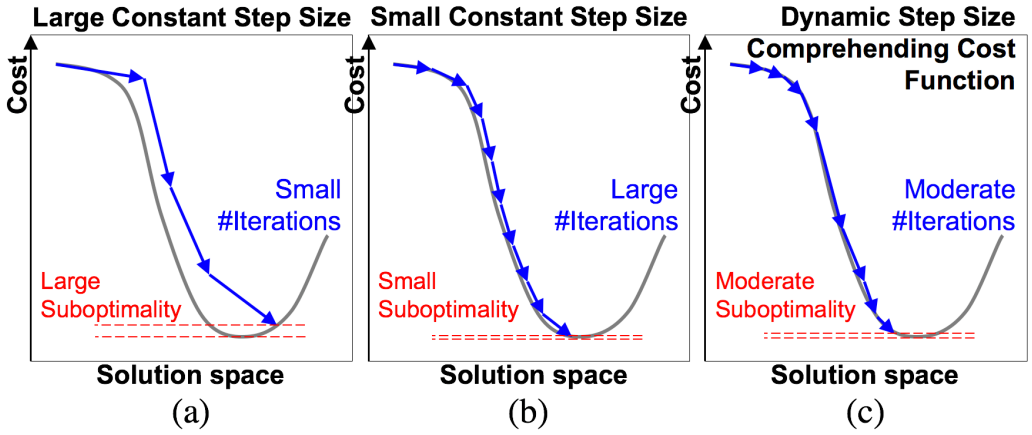


Figure 3.5: An illustration conceptually showing the benefit of dynamic step size adaptation. Cost is composite of wirelength and density. (a) Constant large step size; (b) Constant small step size; and (c) Our improved dynamic step size adaptation.

allocate optimization effort, we dynamically adjust both the step size and the step size scaling range during global placement, while previous approaches [18] [19] [79] [80] [81] adjust step size within a constant step size range.

3.4.1 Improved Dynamic Step Size Adaptation

The density penalty factor λ (in Equation (3.1)) is used to resolve instance overlaps. In previous ePlace implementations [79] [80] [81], the λ scaling tries to maintain a constant HPWL increment[12] across iterations. Algorithm 4 summarizes the ePlace λ scaling methodology, whereby step size (i.e., cof in Algorithm 4) is dynamically scaled to maintain a constant HPWL increment of $\Delta HPWL_{ref}$ (Line 4 in Algorithm 4). In our implementation, we use the same $\Delta HPWL_{ref}$ as in ePlace-MS [81]. Based on HPWL increment per iteration, cof varies within a predefined *step size scale* $[cof_{min}, cof_{max}]$, cof_{max} and cof_{min} respectively indicate the maximum and the minimum step sizes. A larger (resp. smaller) HPWL increment corresponds to a smaller (resp. larger) cof. The previous work of ePlace-MS [81] fixes $cof_{min} = 0.95$ and $cof_{max} = 1.05$.

[12] A "constant cell displacement" results in failure to converge for at least four of 16 (ISPD-2005 and ISPD-2006) testcases; we therefore believe that this is an enabling difference in RePlAce.

Algorithm 4 λ Scaling

1: **Procedure** $\lambda\_Scaling()$
2: $\lambda \leftarrow (\sum_i gradient\_wirelength)/(\sum_i gradient\_potential)$;
3: **for** $k = 0$ to $last\_iteration$ **do**
4: $p \leftarrow (HPWL_k - HPWL_{k-1}) \,/\Delta HPWL_{ref}$;
5: **if** $p < 0$ **then**
6: $cof \leftarrow cof_{max}$;
7: **else**
8: $cof \leftarrow \max(cof_{min},\ pow(cof_{max}, 1-p))$;
9: **end if**
10: $\lambda \leftarrow \lambda \times cof$;
11: **end for**

In RePlAce, to efficiently allocate the optimization effort we propose an *improved dynamic step size adaptation* strategy that dynamically adjusts the step size cof and the maximum step size cof_{max} (i.e., the range of step size scaling) with respect to the HPWL increment per each iteration. Figure 3.6 shows the HPWL curve from the testcase *ADAPTEC1* [138] across iterations in the placement procedure. The slope of the HPWL curve can change at each iteration, and changes rapidly near the star symbols. In our experience, all observed HPWL curves from a wide range of testcases have very strong commonality, with trajectory shapes as shown in Figure 3.6 and two classes of extreme points. We define two classes of *transition points* based on the HPWL curve's instantaneous slope-change rate: (i) *2nd-order transition point* (TP^2), and (ii) *1st-order transition point* (TP^1).

We perform a *trial placement* ahead of the actual placement procedure to capture transition points. We observe that the HPWL on transition points from the trial placement are close to those from the actual global placement procedure that applies our improved dynamic step size adaptation method (i.e., $< 5\%$). The trial placement procedure is described in Section 3.4.2. On the HPWL curve from the placement procedure, the TP^2 points are defined to be the two points with largest absolute instantaneous rate of slope change (red and blue stars in Figure 3.6). Two TP^2 points divide the HPWL curve into three *phases* (blue, green, and yellow regions in Figure 3.6), and each phase has one TP^1 point. The TP^1 point within a given phase is determined as follows. (i) Within each of the 1st and 3rd phases, the TP^1 point has the same instantaneous slope as the line segment

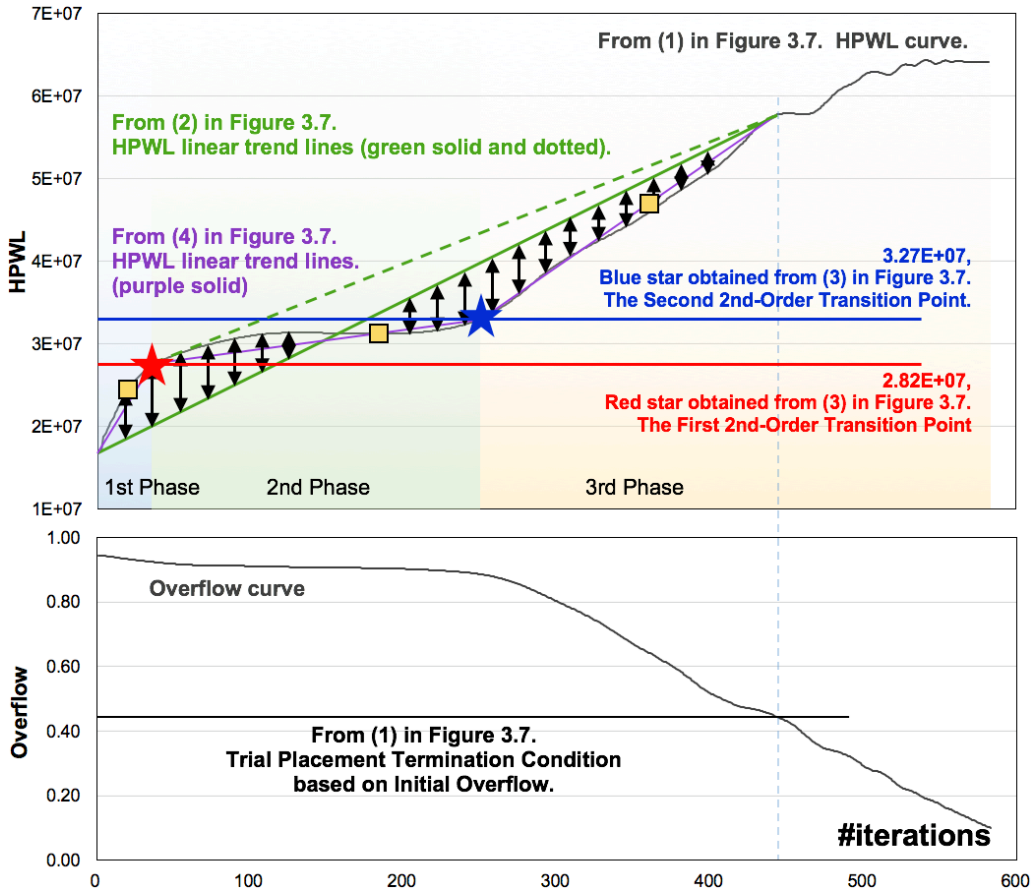


Figure 3.6: HPWL curve of *ADAPTEC1* from the trial placement procedure and the estimated transition points (TP^2 = red/blue stars, TP^1 = yellow squares).

(purple solid line segment in Figure 3.6) drawn between the two extreme (i.e., leftmost and rightmost) points of the HPWL curve within the phase. (ii) Within the 2nd phase, the TP^1 point corresponds to the intersection between the HPWL curve and the purple solid line segment. The TP^1 points are shown as yellow squares in Figure 3.6.

We observe that transition points are important to obtaining improved solution quality because (i) instances tend to alter their moving directions and to settle their locations near the TP^2s, which determines the overall solution quality; and (ii) instances move actively toward their final locations near the TP^1s, which provides an opportunity to save runtime. We allocate optimization effort based on these observations, applying the smallest step size at the TP^2s and the largest step size at the TP^1s. We achieve this

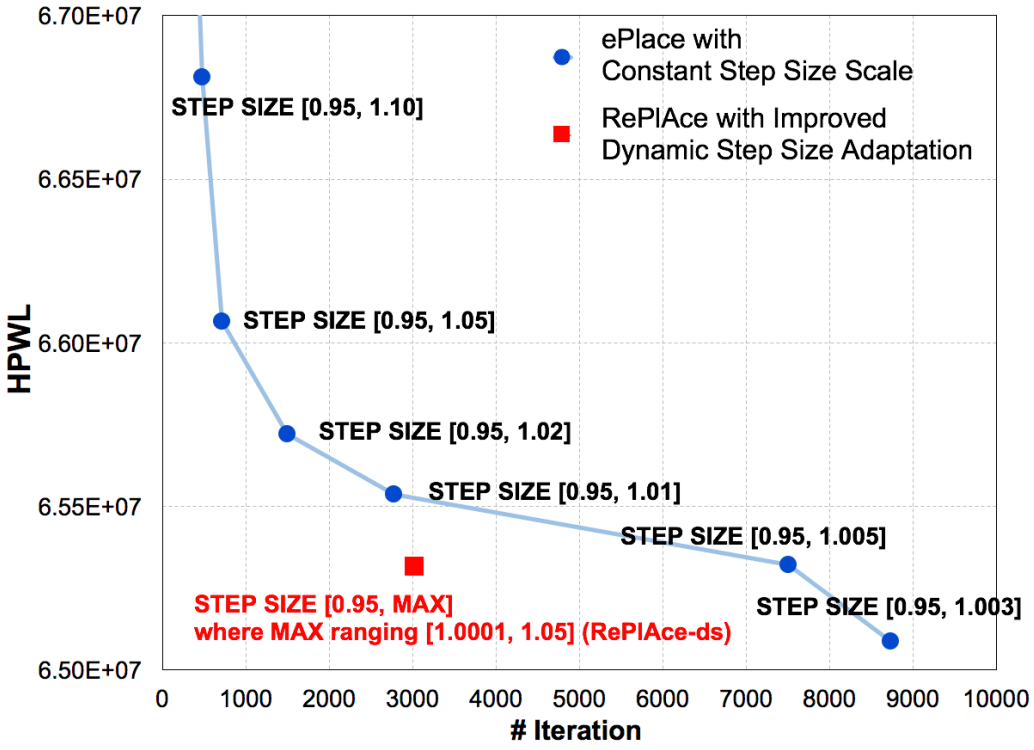


Figure 3.7: Solution qualities achieved by constant step size scale as in ePlace-MS [81] and by RePlAce-ds' improved dynamic step size adaptation strategy, on the *ADAPTEC1* [138] testcase. RePlAce-ds achieves a dominating runtime and solution quality (red square).

dynamical adaptation of the step size scale by controlling the maximum step size cof_{max}. Our empirically determined step size scale ranges from [0.95, 1.0001] to [0.95, 1.05].

$$cof_{max} = min_{cof_{max}} + cof_{range} \times \frac{|HPWL_{current} - HPWL_{TP^2}|}{|HPWL_{TP^1} - HPWL_{TP^2}|} \tag{3.10}$$

With the given $min_{cof_{max}}$, cof_{range}, and the current iteration's HPWL, we compute maximum step size using Equation (3.10), which achieves dynamic control of the step size scale. In our implementation, $min_{cof_{max}}$ is 1.0001 in the first phase, 1.001 in the second phase, and 1.005 in the third phase. We empirically determine cof_{range} as 0.0009, 0.024 and 0.045 for the three phases, respectively. We allocate more optimization effort at the beginning of the nonlinear optimization based on our observations that the solution quality achieved in early iterations is critical to the final placement solution. Figure 3.7 shows the

tradeoff between solution quality (HPWL) and runtime (#iterations) with various *constant* step sizes. Our improved dynamic step size adaptation achieves a superior result (red square) in terms of both HPWL and runtime. We note that ePlace-MS (blue line) does not converge with smaller step sizes (i.e., $[0.95, 1.002]$ or below).

3.4.2 Trial Global Placement

To find the 1st-order (TP^1) and the 2nd-order (TP^2) transition points on the HPWL curve, we perform a *trial placement procedure* that includes a trial global placement (tGP). We terminate tGP when the density overflow is $\leq \frac{\tau_{init}}{2.5}$.[13] Figure 3.8 describes our trial procedure. Inspired by [108], the transition points are determined as follows. (1) In Line 2 of Algorithm 5, we first perform tGP and obtain the HPWL curve, as shown in Figure 3.6. (2) In Line 3, we connect the initial (start of tGP) and final (end of tGP) HPWL points on the HPWL curve by a *primary* line segment (green solid line segment in Figure 3.6). In Line 4, for each tGP iteration (along the x-axis in Figure 3.6), we calculate the absolute HPWL differences between the HPWL curve and the primary line segment (black arrows in Figure 3.6). (3) In Line 5, we pick the point with the largest absolute difference as one of the TP^2 points (red star in Figure 3.6). (4) To find the other TP^2 point, we connect the existing TP^2 point to the initial and final HPWL points, respectively, using two *secondary* line segments (green dotted line segments in Figure 3.6). The point (blue star in Figure 3.6) with the largest absolute HPWL difference between the HPWL curve and the two secondary line segments is determined as the other TP^2 point. (5) In Line 6, we divide the HPWL curve into three phases based on the two TP^2 points. In Line 7, we repeat (2) separately for the HPWL curve of each phase. (6) In Lines 8-9, we repeat (3) to find a TP^1 point within each of the 1st and 3rd phases. The TP^1 point in the 2nd phase is the point of intersection where the HPWL curve transitions from above to below the purple line segment.

[13] Overall density overflow τ is defined as the sum of the density overflow for all placement bins over the total cell area. $0 \leq \tau \leq 1$. The initial density overflow τ_{init} is the density overflow obtained from a wirelength-only optimization before our nonlinear optimization. We empirically determine the constant as 2.5, which provides better results in terms of tradeoff between HPWL and the number of iterations.

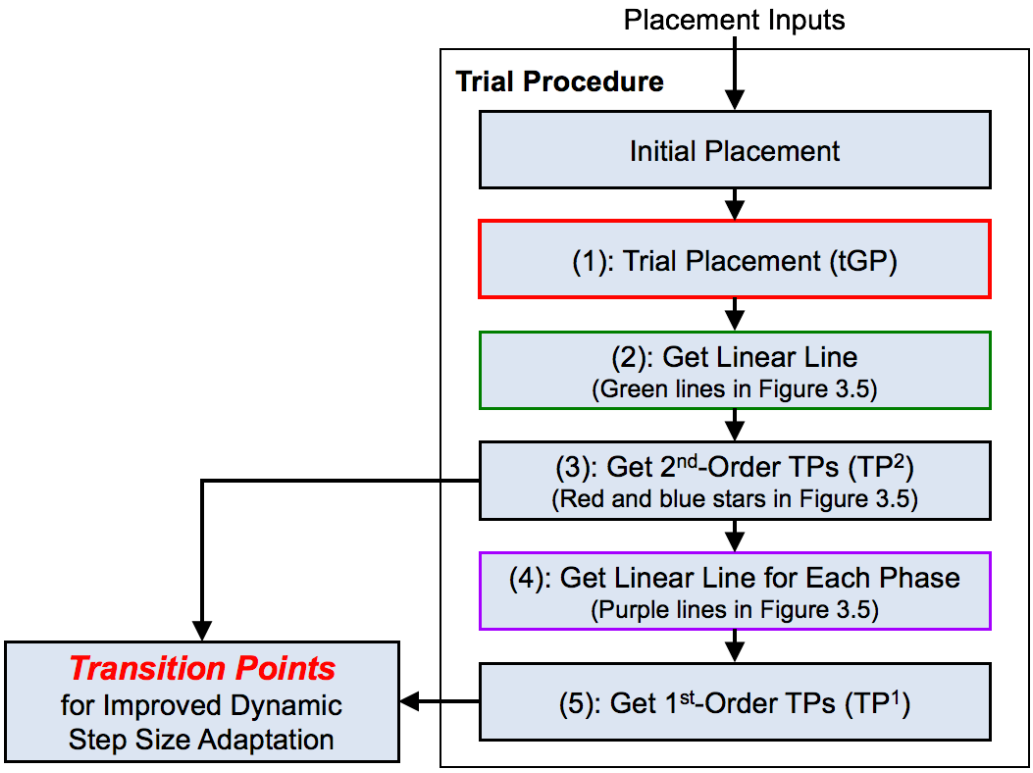


Figure 3.8: Flowchart of our trial placement procedure. The red rectangle indicates non-linear optimization using Nesterov's method. The actual placement procedure follows this trial placement procedure.

Algorithm 5 Finding Transition Points

Procedure $Trial()$
Trial placement();
Connect initial and final points on HPWL curve using a linear line;
Calculate HPWL differences between HPWL curve and linear line for each tGP iteration;
Get the 2nd-order transition points (TP^2);
Divide the HPWL curve into three phases by TP^2;
Connect initial and final points on HPWL curve separately for each phase;
Calculate HPWL differences between HPWL curve and linear line for each phase for each tGP iteration;
Get the 1st-order transition points (TP^1);
return TP^2 and TP^1;

3.5 Routability-Driven Placement

To produce routable placement has recently being recognized as being of critical importance. This is reflected by many routability-driven placement contests [11] [120] [121] [122] [146] and placers [24] [27] [43] [47] [66] [75]. In this section, we describe how global routability-driven placement is achieved in RePlAce, including (i) capacity and blockage calculation; (ii) demand calculation; (iii) cell inflation technique; and (iv) overall flow. Notations are described in Table 3.2.

3.5.1 Capacity and Blockage Calculation

According to DAC-2012 [121] and ICCAD-2012 [122] contest and benchmark suite descriptions, global routing is performed using global routing tiles, with back-end-of-line capacity (i.e., tile width or height) and blockage (in the unit of tile dimension) defined for each layer in the official benchmark inputs. We follow the definitions from [121] [122], and associate a capacity value (*cap*) to each edge of the global routing tile for each metal layer. As an example, for a unidirectional horizontal routing layer, the left and right tile edges have a capacity value directly from the benchmark specification, while the up-

Table 3.2: Notations for routability-driven placement. We use the default length unit in the DAC-2012 and ICCAD-2012 benchmark suites.

Term	Description
blk	blocked routing capacity per tile edge
cap	routing capacity (by length unit) per tile edge
$demand$	routing demand per tile edge
$infl\_ratio$	cell inflation ratio
ml	metal layer index
$pincnt$	pin count per tile
$pitch_{ml}$	metal pitch (by length unit) on layer ml
rt	global routing tile index
$usage$	# tracks used per tile edge

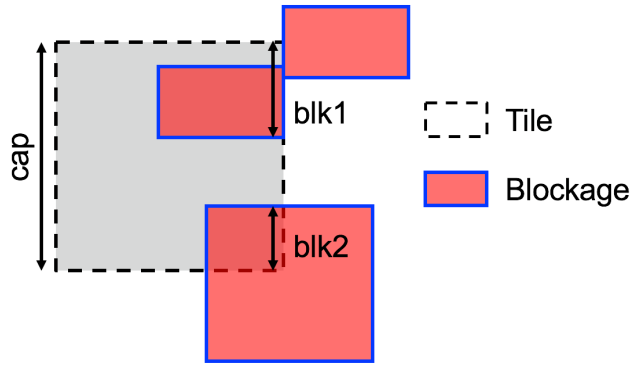


Figure 3.9: Illustration of blockage calculation. For the vertical edge on the right, $blk = blk1 + blk2$. Note the union of blocked capacity for the upper two blockages.

per and lower tile edges have a capacity of zero. We also associate a routing blockage (*blk*) to each edge of a global routing tile *for each metal layer*. The blockage is defined as the total blocked capacity (total blocked tile width or height), as illustrated in Figure 3.9.

3.5.2 Demand Calculation

During global placement, we invoke the official router *NCTU-GR* [161] from the DAC-2012 and ICCAD-2012 contests to obtain the routing demand. As described in [121], the global router reports cross-tile routing segments, so that tile edge-based routing usage (#tracks used) can be obtained. For each layer, we multiply the usage (*usage*) with metal pitch to obtain the routing demand. Additionally, for metal 2 and below, we further consider the pin blockage effect. Here, we calculate the total number of pins (*pincnt*) within a tile and use γ_{pin} as a pin blockage factor. We describe the demand calculation in Equation (3.11):

$$demand_{e,ml} = (usage_{e,ml} + \gamma_{pin} \cdot pincnt) \cdot pitch_{ml}, \tag{3.11}$$

where the subscript e indicates one of the four edges of a given global routing tile, and the subscript *ml* indicates a specific metal layer. Thus, we calculate the demand for all four tile edges and for all metal layers. We use the same pin blockage factor as specified in each of the benchmark suites, i.e., $\gamma_{pin} = 0$ for DAC-2012 benchmark and $\gamma_{pin} = 0.05$ for ICCAD-2012 benchmark.

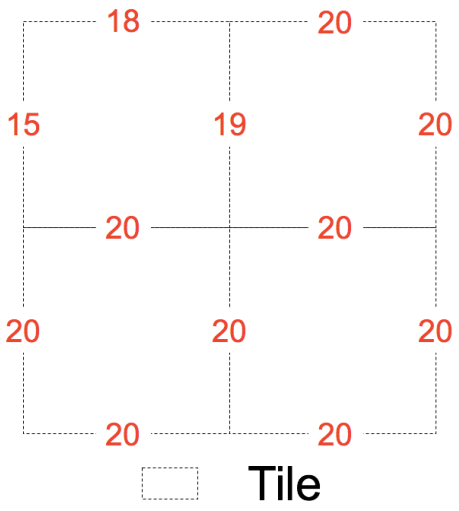


Figure 3.10: An example of routing demand calculation: the upper-left tile has a horizontal routing demand of $\max(15,19) = 19$, and a vertical routing demand of $\max(18,20) = 20$.

3.5.3 Cell Inflation

To resolve congestion, we inflate cells within tiles for which the demand is larger than the corresponding capacity. Since we calculate the capacity and demand per tile edge on each layer, there are multiple (capacity, demand) pairs. The inflation ratio for each cell is calculated as the maximum demand over capacity ratio, as shown in Equation (3.12). Cell width and height are increased according to the square root of the inflation ratio for each direction. Based on empirical studies, we enable superlinear cell inflation with $\gamma_{super} = 2.33$ and we bound the maximum inflation ratio to be 2.5. In this way, we achieve metal layer-aware inflation, rather than relying only on sums of capacities and demands over all metal layers [43] [75].

$$infl\_ratio = \max_{all\ e,ml} \left(\left(\frac{demand_{e,ml} + blk_{e,ml}}{cap_{e,ml}} \right)^{\gamma_{super}}, 2.5 \right) \tag{3.12}$$

To avoid the total inflated area exceeding available white-space, we adopt the dynamic inflation ratio adjustment methodology from [42]. Algorithm 6 describes the inflation ratio adjustment. We first calculate the total inflated area according to the initial

Algorithm 6 Inflation Ratio Adjustment

Procedure $AdjustInflationRatio$
$total\_inflated\_area \leftarrow GetTotalInflatedArea()$;
while $total\_inflated\_area \geq max\_inflated\_area$ **do**
  $infl\_ratio_0 \leftarrow GetInflationRatioForLeastCongestedTile()$;
  for all $tiles$ (rt) **do**
    $infl\_ratio_{rt} \leftarrow \frac{infl\_ratio_{rt}}{infl\_ratio_0}$;
  end for
  $total\_inflated\_area \leftarrow UpdateTotalInflatedArea()$;
end while

Algorithm 7 Routability Optimization

Procedure $RouteOpt$
$CalculateInflationRatio()$;
$AdjustInflationRatio()$;
$InflateCell()$;
$\lambda_{curr\_iter} \leftarrow \lambda_{curr\_iter-100}$;
$AdjustUtilization()$;
$UpdateEarlyTerminationIndicator()$;

inflation ratio. If the total inflated area exceeds a predefined maximum value, $max\_inflated\_area$, we divide the horizontal inflation ratio for each tile by the inflation ratio of the least-congested tile that has a ratio greater than one, and recalculate the total inflated area. We repeat the above procedure until the total inflated area becomes smaller than the predefined maximum value. We describe $max\_inflated\_area$ in Section 3.5.4.

3.5.4 Overall Flow

Figure 3.11 shows the overall flow of our global routability-driven placement. When the wirelength-driven global placement reaches 20% density overflow, we invoke the global router *NCTU-GR* [161] to obtain our internal routing congestion (RC) evaluation. We then perform routability optimization using our cell inflation technique (see discussion below), then feed the new cell sizes, die utilization, and density penalty factor back to the global placement engine. We choose to perform routability optimization only if we meet all three of the following conditions: (i) the latest congestion estimation indicates a less than 1% routing overflow ($RC < 1.01$); (ii) fewer than 10 rounds of cell inflation have been performed; and (iii) the binary indicator *earlyTermination* is false.

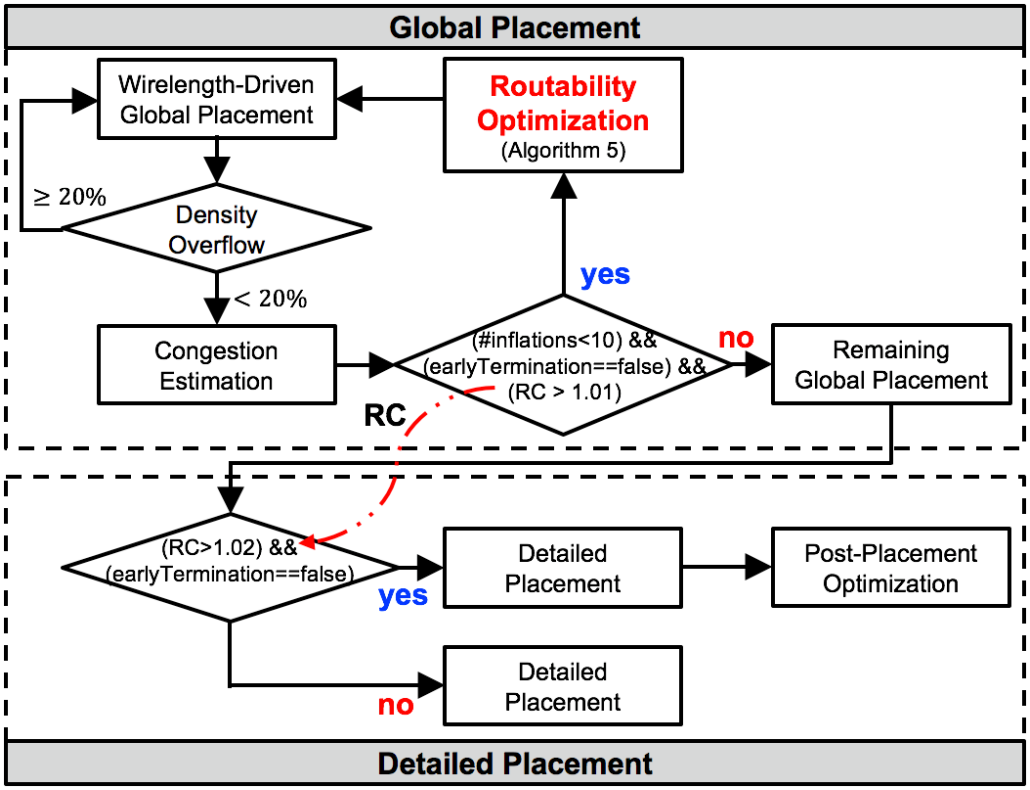


Figure 3.11: Overall flowchart of our routability-driven placement.

Otherwise, we execute the remaining global placement procedure and pass the placement solution on to the detailed placer. The binary *earlyTermination* flag is used to skip the following rounds of cell inflation once the design is considered difficult to improve further (see discussion below). Since our work solely focuses on global placement, we use *NTUplace3* [19] as the detailed placer. We then perform a post-placement optimization [76] if we believe there can be further benefits. In our implementation, we go through post-placement optimization only when both of the following conditions are met: (i) there is still more than 2% routing overflow ($RC > 1.02$); and (ii) there is still room for improvement (*earlyTermination* is false). Results reported below (in Section 3.6.3) show that improvements over previous works can be attributable to RePlAce global placement as opposed to the use of *NTUplace3 (or, even,* NTUplace4h *[47]).*

The routability optimization process is described in Algorithm 7. In Line 2, we first calculate the inflation ratio from congestion estimation, as described in Sections 3.5.1

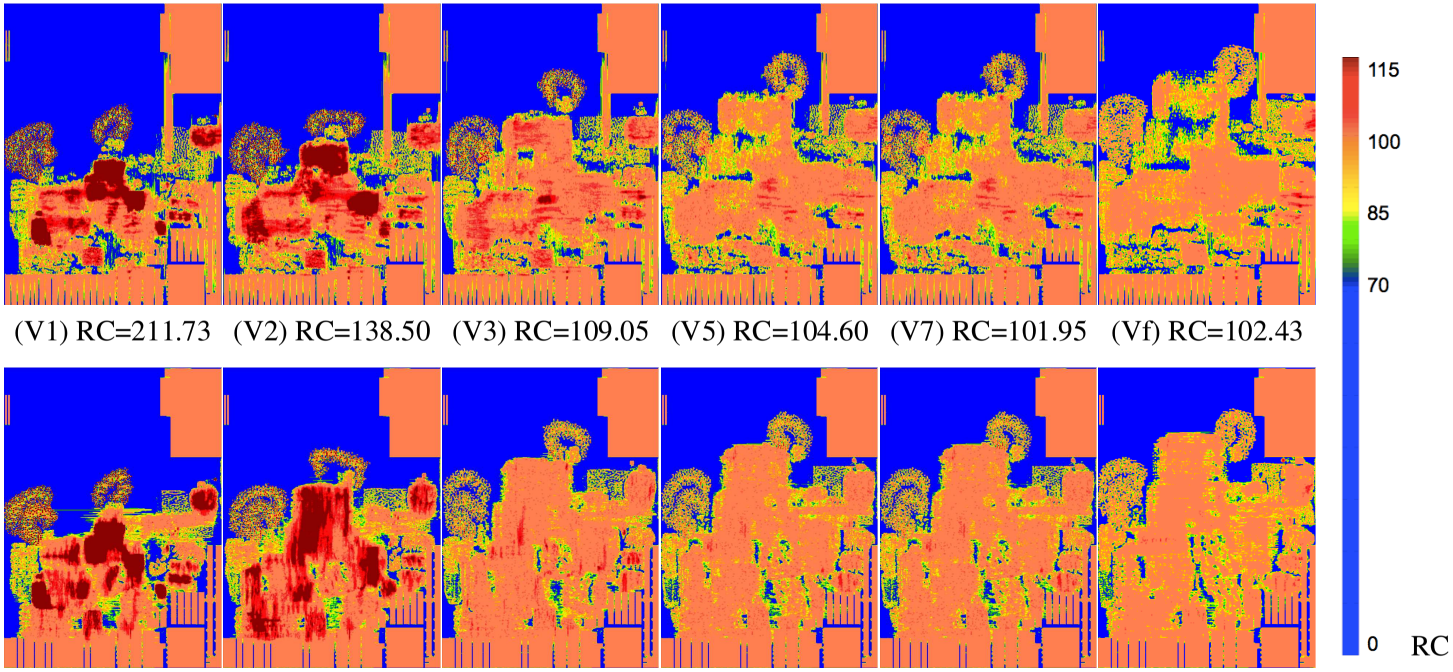
and 3.5.2. Then, in Line 3, we adjust the inflation ratio according to Algorithm 6. In Line 4, we perform cell inflation to improve routability. In Line 5, we roll back the density penalty factor λ by 100 Nesterov's optimization iterations [81] to encourage re-placement of cells based on the new cell sizes. Line 6 reflects that the overall die utilization should be adjusted because the equivalent total cell area becomes larger due to inflation. We adjust the die utilization based on the ratio of equivalent total cell area over die area, as given in Equation (3.13). In each routability optimization, we limit $max\_inflated\_area$ to be 10% of the total whitespace area, so that the total utilization is constrained to remain less than 100%. In Line 7, we update the binary indicator $earlyTermination$. The indicator remains false until the minimum RC value thus far has not improved by 0.008 over the last four consecutive rounds of cell inflation.

$$util = \frac{curr\_cell\_area + total\_inflated\_area}{die\_area} \tag{3.13}$$

Figure 3.12 shows snapshots of congestion maps during our routability optimization procedure for *SUPERBLUE12*. The figure shows how the global placement process effectively reduces the congestion (i.e., hotspots) indicated by red regions. In Figure 3.12, RC = Routing Congestion reported by the DAC-2012 official evaluation script. Figures (Hk) and (Vk) show horizontal and vertical congestion before the k^{th} cell inflation. Figures (Hf) and (Vf) show final horizontal and vertical routing congestion after detailed placement.

Table 3.3: RePlAce functionalities and the corresponding suffixes (command-line options) that produce the results reported below.

Benchmark Type	-ld	-ds	-ldds	-r
ISPD-2005 and ISPD-2006 benchmarks		•		
MMS benchmarks	•	•	•	
DAC-2012 and ICCAD-2012 benchmarks				•



(V1) RC=211.73 (V2) RC=138.50 (V3) RC=109.05 (V5) RC=104.60 (V7) RC=101.95 (Vf) RC=102.43

(H1) RC=211.73 (H2) RC=138.50 (H3) RC=109.05 (H5) RC=104.60 (H7) RC=101.95 (Hf) RC=102.43

Figure 3.12: Global routing overflow (*SUPERBLUE12*) during routability-driven global placement procedure.

3.6 Experiments

In this section, we describe our experimental setups and results. We implement RePlAce in C++ and perform experiments in single-thread mode using a 2.5GHz Intel Xeon server. Our implementation has no benchmark-specific code or tuning: a single binary produces all results, with command-line options as we describe below. Experiments are performed on three types of well-studied academic benchmarks: (i) ISPD-2005 [93] and ISPD-2006 [92] benchmark suites for standard cell placement; (ii) the large-scale modern mixed-size (MMS) [138] benchmark suite for mixed-size placement; and (iii) DAC-2012 [121] and ICCAD-2012 [122] benchmark suites for global routability-driven placement. RePlAce functionalities and corresponding suffixes (command-line options) are summarized in Table 3.3. We briefly give some insight into the remaining gaps between academic and real-world placers and testcases, by comparing final-routed wirelength for real standard-cell placements obtained by RePlAce-r and a leading-edge commercial place-and-route tool in a foundry 28LP technology. In all of our experimental results tables, bold numbers indicate the best HPWL (sHPWL) for each testcase.

3.6.1 Standard Cell Placement

For standard cell placement, we validate RePlAce using the ISPD-2005 [93] and ISPD-2006 [92] benchmark suites, whose parameters are summarized in Table 3.4. We employ *NTUplace3* [19] as our detailed placer. Experimental results are summarized in Table 3.5. For testcases with a specified target density, we report the scaled HPWL using the official evaluation scripts [92].

Table 3.5 compares RePlAce-ds to the best known results [80] [150] across ISPD-2005 and ISPD-2006 benchmark suites. In Table 3.5, best known ISPD results are cited from Tables II and V of Nonsmooth placer [150] and Tables II and V of ePlace [80].

We observe that RePlAce-ds achieves (new) best known results for 15 out of 16 testcases. Overall, RePlAce-ds achieves up to 4.00% HPWL reduction (*ADAPTEC2*) and 2.00% HPWL reduction on average, when compared to the previous best known results.

Table 3.4: Statistics for ISPD-2005 [93], ISPD-2006 [92], and MMS [138] benchmark suites.

Circuits	# Objects	# Standard Cells	# Nets	Target Density (%)	ISPD Macros		MMS Macros	
					#Movable	#Fixed	#Movable	#Fixed
ADAPTEC1	211447	210904	221142	100	0	543	63	480
ADAPTEC2	2255023	254457	266009	100	0	566	127	439
ADAPTEC3	2451650	450927	466758	100	0	723	58	665
ADAPTEC4	2496045	494716	515951	100	0	1329	69	1260
BIGBLUE1	2278164	277604	284479	100	0	560	32	528
BIGBLUE2	2557866	534782	577235	100	0	23084	959	22125
BIGBLUE3	21096812	1093034	1123170	100	2485	1293	2549	1229
BIGBLUE4	22177353	2169183	2229886	100	0	8170	199	7970
ADAPTEC5	2843128	842482	867798	50	0	646	76	570
NEWBLUE1	2330474	330137	338901	80	64	337	64	337
NEWBLUE2	2441516	436516	465219	90	3723	1277	3748	1252
NEWBLUE3	2494011	482833	552199	80	0	11178	51	11127
NEWBLUE4	2646139	642717	637051	50	0	3422	81	3341
NEWBLUE5	21233058	1228177	1284251	50	0	4881	91	4790
NEWBLUE6	21255039	1248150	1288443	80	0	6889	74	6815
NEWBLUE7	22507954	2481372	2636820	80	0	26582	161	26421

Table 3.5: (Scaled†) HPWL ($\times 10^6$) and runtime (minutes) comparison of best known, RePlAce-ld, RePlAce-ds and RePlAce-ldds on ISPD and MMS benchmarks.

Circuits	ISPD-2005, ISPD-2006				MMS							
	Best known [80] [150]		RePlAce-ds		Best known [81]		RePlAce-ld		RePlAce-ds		RePlAce-ldds	
	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU
ADAPTEC1	74.20	13.13	**73.01**	14.18	66.82	5.47	65.17	5.30	65.32	15.34	**64.98**	19.08
ADAPTEC2	84.84	4.90	**81.45**	25.75	76.74	7.58	72.75	7.68	71.68	20.43	**71.51**	25.08
ADAPTEC3	194.07	26.13	**190.45**	50.88	161.55	27.23	154.18	23.22	**151.34**	62.81	151.42	69.15
ADAPTEC4	174.11	27.85	**172.22**	87.55	145.89	56.40	142.05	32.91	**139.70**	96.71	140.57	117.35
BIGBLUE1	90.98	6.25	**89.05**	23.78	86.29	7.82	85.79	8.15	86.03	23.85	**85.04**	28.23
BIGBLUE2	141.83	10.50	**136.67**	48.19	130.06	13.70	**125.33**	16.29	125.84	47.95	125.49	53.46
BIGBLUE3	306.94	27.29	**298.61**	112.98	284.39	72.98	**270.17**	73.83	282.42	165.23	280.31	183.76
BIGBLUE4	742.45	145.00	**740.57**	337.23	656.68	204.15	653.24	162.34	650.09	317.67	**647.55**	363.32
ADAPTEC5†	**391.02**	83.65	391.24	74.92	**294.24**	46.07	303.36	35.13	301.78	81.83	302.53	92.53
NEWBLUE1†	59.26	14.00	**57.44**	27.56	60.43	11.70	58.63	9.90	57.75	27.56	**57.44**	31.67
NEWBLUE2†	182.42	20.01	**177.82**	32.56	159.11	51.12	152.32	15.44	152.34	51.31	**150.09**	58.29
NEWBLUE3†	264.48	33.15	**255.07**	68.62	269.47	36.30	258.53	20.97	**257.22**	57.19	257.67	65.76
NEWBLUE4†	269.58	56.26	**267.71**	58.33	226.29	28.27	**223.52**	26.08	224.02	59.92	223.62	68.24
NEWBLUE5†	492.62	54.83	**486.37**	118.19	392.77	55.47	390.14	75.81	388.74	151.55	**386.30**	167.19
NEWBLUE6†	464.36	116.70	**460.24**	118.45	409.28	69.65	408.89	84.04	407.04	168.23	**406.60**	184.09
NEWBLUE7†	978.07	246.00	**950.27**	335.19	889.18	392.02	**876.36**	172.58	877.83	277.77	880.67	326.00
Avg.	+0.00%	1.00×	-2.00%	1.78×	+0.00%	1.00×	-2.25%	0.72×	-2.43%	1.81×	-2.73%	2.09×

3.6.2 Mixed-Size Placement

We demonstrate the benefit from our constraint-oriented local density function using the large-scale modern mixed-size (MMS) placement benchmarks [138]. Parameters of the benchmarks are summarized in Table 3.4. The MMS benchmarks embody the same designs as the ISPD-2005 and ISPD-2006 benchmarks, except that some macros are movable. We use *NTUplace3* [19] as our detailed placer. Experimental results are summarized in Table 3.5. For testcases with a specified target density, we report scaled HPWL using the official evaluation scripts [138].

Table 3.5 compares RePlAce mixed-size placement results with best known previous results, referred from Table II of ePlace-MS [81]. We apply three different schemes: (1) local density function equipped RePlAce-ld,[14] (2) improved dynamic step size adaptation equipped RePlAce-ds, and (3) combined RePlAce-ldds. Runtimes of (2) and (3) include the trial procedure's runtimes. sHPWL is HPWL scaled by placement density overflow. Runtime with "*" indicates cited results. All cited and reported results use *NTUplace3* [19] as the detailed placer. From the bottom row of the table, we see that RePlAce-ld reduces HPWL by approximately 2.25% on average compared to best known results.[15] Compared to best known results, RePlAce-ld produces shorter HPWL for 15 out of 16 testcases. In addition, RePlAce-ldds shows further improvement of solution quality with the addition of dynamic step size adaptation. Albeit with increased runtime, RePlAce-ldds appears to effectively invest its effort (i.e., iterations, runtime), and achieve the best solution quality on average, by incorporating both the -ld and -ds techniques. Together, RePlAce-ds and RePlAce-ldds produce the best solution quality for 15 out of 16 testcases. As our results show merits to both constraint-oriented local smoothing and dynamic step size adaptation, as well as their explicit combination, we leave to future practitioners the question of how to best apply and orchestrate these techniques.

Last, Figure 3.13 shows a breakdown of #iterations across the component global placement procedures for (a) RePlAce-ds with the ISPD-2005 and ISPD-2006 benchmark suites, and (b) RePlAce-ds and (c) RePlAce-ldds with the MMS benchmark suite, aggre-

[14] We see little benefit by applying local density to testcases without large movable macros. Thus, "-ld" option is only applied to MMS benchmark suites.

[15] However, with local density function, we do not find a uniform trend with accelerated convergence rate for global placement. The local density calculation takes on average $\sim$ 1.8% of the total runtime.

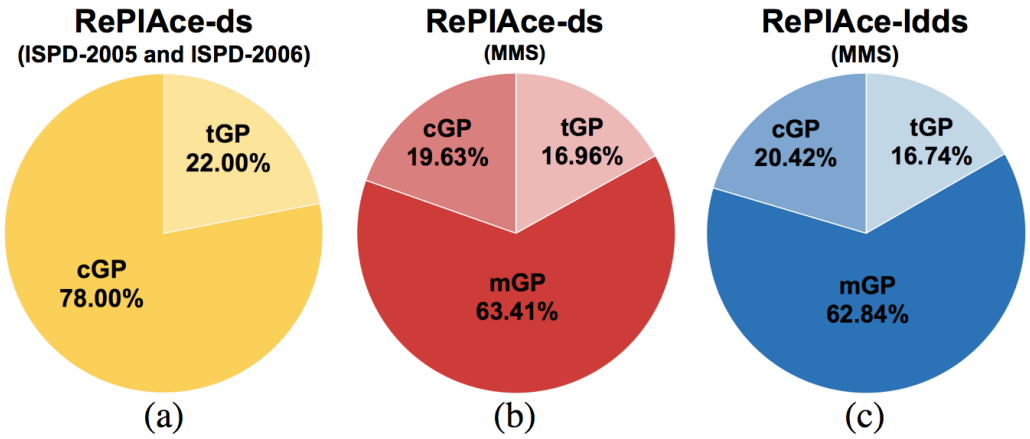


Figure 3.13: Runtime breakdown (#iterations) aggregated over all reported testcases in the ISPD-2005 and ISPD-2006 benchmark suites for (a) RePlAce-ds, and in the MMS benchmark suite for (b) RePlAce-ds and (c) RePlAce-ldds.

gated over all reported testcases. (Note that for any given testcase, runtime will be roughly proportional to #iterations.) Here, tGP, mGP, and cGP respectively denote trial placement, macro and standard cell placement, and standard cell-only placement. In the figure, tGP indicates trial placement; mGP indicates macro and standard-cell placement; and cGP indicates standard-cell only placement. With dynamic step size adaptation, approximately 17-22% of iterations (i.e., of runtimes) are additionally consumed by the tGP procedure.

3.6.3 Routability-Driven Placement

We validate RePlAce global routability-driven placement using the DAC-2012 [121] and ICCAD-2012 [122] benchmark suites. We compare our placement solutions to those of all published results from leading-edge academic placers [24] [43] [47] [75]. Parameters of the DAC-2012 and ICCAD-2012 benchmark suites are summarized in Table 3.6. The DAC-2012 and ICCAD-2012 benchmark suites do not include movable macros, but contain *.shapes* and *.route* files that respectively describe the component shapes for non-rectangular nodes and routing-related information (e.g., the number of routing tracks per each metal layer, routing blockages, etc.).

Experimental results are summarized in Table 3.7 and Table 3.8. Table 3.7 shows scaled HPWL (sHPWL) ($\times 10^7$), routing congestion (RC) and CPU runtime (minutes) comparison of RePlAce to leading published results for DAC-2012 [121] routability-

driven placement. sHPWL is HPWL scaled with routing congestion penalty. (Each unit of RC in excess of 100 results in a 3% sHPWL penalty.) Published results are cited from best contest results at DAC-2012 [121], and newest versions of mPL [24]. Missing benchmark results are indicated by N/A. "RePlAce-r alt" uses *NTUplace4h* as its detailed placer. Global routing is performed by the official global router *NCTU-GR* [161], and sHPWL is evaluated by the official DAC-2012 contest evaluation scripts. To match the reporting convention in Table 3.5, the improvement of RePlAce-r over previous best known DAC-2012 results (with those previous best known results set to be 1.00×) would be -7.93%. Runtime with "*" indicates cited results. Only *SUPERBLUE2* invokes post-placement optimization, with sHPWL improvement of 2.74% (sHPWL from 62.58 to 60.87). Table 3.8 shows scaled HPWL (sHPWL) ($\times 10^7$), routing congestion (RC) and CPU runtime (minutes) comparison of RePlAce to leading published results for ICCAD-2012 [122] routability-driven placement. sHPWL is HPWL scaled with routing congestion penalty. (Each unit of RC in excess of 100 results in a 3% sHPWL penalty.) Published results are cited from best contest results at ICCAD-2012 [122], and newest versions of POLAR 2.0 [75], NTUplace4h [47] and Ripple 2.0 [43] (listed in chronological order). "RePlAce-r alt" uses *NTUplace4h* as its detailed placer. Global routing is performed by the official global router *NCTU-GR* [161], and sHPWL is evaluated by the official ICCAD-2012 contest evaluation scripts. To match the reporting convention in Table 3.5, the improvement of RePlAce-r over previous best known ICCAD-2012 results (with those previous best known results set to be 1.00×) would be -6.66%. Runtime with "*" indicates cited results. Only *SUPERBLUE10* invokes post-placement optimization, with sHPWL improvement of 7.47% (sHPWL from 63.12 to 58.41).

For all testcases, we set the target density as 90%[16]. We report scaled HPWL (sHPWL) and routing congestion (RC) as evaluated by the official global router *NCTU-GR* [77] [161] and contest evaluation scripts [121] [122]. In the DAC-2012 and ICCAD-2012 benchmark suites, routing congestion is described using the "ACE(X)" metric, which is the averaged congestion of the top X% tile edges. Then, peak weighted congestion (PWC) and Routing Congestion (RC) are obtained by Equations (3.14) and (3.15), where

[16]In our work, we apply the uniform 90% target density, based on experience, to balance wirelength and congestion from a initial global placement perspective, and we keep using the same value without per-benchmark tuning.

$k_i = 1$. Equation (3.16) defines the final evaluation metric, i.e., scaled HPWL. Each unit of RC in excess of 100 results in a 3% sHPWL penalty.

$$PWC = \frac{\Sigma_{i=0.5,1,2,5} k_i \cdot ACE(i)}{\Sigma k_i} \tag{3.14}$$

$$RC = \max(100, PWC) \tag{3.15}$$

$$sHPWL = HPWL \times (1 + 0.03 \times (RC - 100)) \tag{3.16}$$

Compared to all published results, RePlAce achieves smaller sHPWL for all testcases (10 and eight testcases from DAC-2012 and ICCAD-2012, respectively). RePlAce achieves on average 9.80% and 9.60% sHPWL reduction over best DAC-2012 and ICCAD-2012 contest results, respectively; and on average 8.50% to 9.59% scaled HPWL reduction over the leading previous academic placers.[17] In our flow as shown in Figure 3.11, 16 out of 18 testcases automatically bypass the post-placement optimization stage, and only two testcases (*SUPERBLUE2* and *SUPERBLUE10*) invoke post-placement optimization, with sHPWL improved by 2.74% and 7.47% respectively. These indicate that our routability-driven global placement effectively reduces overall routing congestion, with an average RC value similar to all published leading-edge results.

To show the impact of the detailed placer, we have conducted additional experiments with the DAC-2012 and ICCAD-2012 benchmark suites by employing a *routability-driven* detailed placer, *NTUplace4h* [47], shown as "RePlAce-r alt" in Table 3.7 and Table 3.8. Interestingly, all 18 testcases end up with worse sHPWL than "RePlAce-r", and 11 out of 18 testcases have worse routing congestion. However, "RePlAce-r alt" is still 7.09% to 8.47% better on average compared to the leading previous academic placers. This provides confirmation of the effectiveness of our routability optimization during the global placement.

[17] To match the reporting convention in Table 3.5, the improvement over previous best known DAC-2012 and ICCAD-2012 results (with those previous best known results set to be 1.00×) would be -7.93% and -6.66%, respectively.

Table 3.6: Statistics for DAC-2012 [121] and ICCAD-2012 [122] benchmark suites.

Circuits	# Objects	# Movable Cells	# Terminal Nodes	# Nets	# Pins	Util (%)	Density (%)
SUPERBLUE1	849441	765102	52627	822744	2861188	69	35
SUPERBLUE2	1014029	921273	59312	990899	3228345	76	28
SUPERBLUE3	919911	833370	55033	898001	3110509	73	42
SUPERBLUE4	600220	521466	40550	567607	1884008	70	44
SUPERBLUE5	772457	677416	74365	786999	2500306	77	37
SUPERBLUE6	1014209	919093	65316	1006629	3401199	73	43
SUPERBLUE7	1364958	1271887	66995	1340418	4935083	76	58
SUPERBLUE9	846678	789064	37574	833808	2898853	73	47
SUPERBLUE10	1202665	1045874	96251	1158784	3894138	70	32
SUPERBLUE11	954686	859771	67303	935731	3071940	79	40
SUPERBLUE12	1293433	1278084	8953	1293436	4774069	56	44
SUPERBLUE14	634555	567840	44743	619815	2049691	72	50
SUPERBLUE16	698741	680450	419	697458	2280931	69	46
SUPERBLUE18	483452	442405	25063	468918	1864306	67	47
SUPERBLUE19	522775	506097	286	511685	1714351	78	49

Table 3.7: Scaled HPWL (sHPWL) ($\times 10^7$), routing congestion (RC) and CPU runtime (minutes) comparison of RePlAce to leading published results for DAC-2012 [121] global routability-driven placement.

Circuits	best in contest [121]			mPL12 [24]			RePlAce-r			RePlAce-r alt		
	sHPWL	RC	CPU	sHPWL	RC	CPU	sHPWL	RC	CPU	sHPWL	RC	CPU
SUPERBLUE2	62.40	100.68	291	61.40	N/A	312	**60.87**	100.96	155	60.96	101.00	160
SUPERBLUE3	36.20	100.56	236	36.00	N/A	215	**30.68**	100.78	62	32.08	102.10	64
SUPERBLUE6	34.25	100.32	186	34.00	N/A	285	**31.20**	100.51	41	31.40	100.52	43
SUPERBLUE7	39.85	100.71	433	39.50	N/A	287	**37.28**	101.22	47	37.36	101.07	51
SUPERBLUE9	25.46	102.48	219	25.00	N/A	212	**21.28**	100.78	42	21.39	100.81	44
SUPERBLUE11	34.22	100.02	254	34.00	N/A	245	**33.69**	102.07	52	34.20	102.35	54
SUPERBLUE12	31.19	100.02	581	30.40	N/A	320	**26.52**	102.43	75	27.49	103.02	80
SUPERBLUE14	22.56	100.07	156	24.50	N/A	126	**21.21**	100.65	16	21.32	100.68	18
SUPERBLUE16	27.39	101.38	46	27.40	N/A	157	**25.27**	101.87	42	25.51	101.94	44
SUPERBLUE19	15.31	100.61	140	15.10	N/A	165	**14.27**	100.71	29	14.65	101.55	30
Avg.	+9.80%	0.995×	5.31×	+9.59%	N/A	5.00×	+0.00%	1.000×	1.00×	+1.54%	1.003×	1.05×

Table 3.8: Scaled HPWL (sHPWL) ($\times 10^7$), routing congestion (RC) and CPU runtime (minutes) comparison of RePlAce to leading published results for ICCAD-2012 [122] global routability-driven placement.

Circuits	best in contest [122]			Polar 2.0 [75]			NTUplace4h [47]		
	sHPWL	RC	CPU	sHPWL	RC	CPU	sHPWL	RC	CPU
SUPERBLUE1	27.89	100.97	39	28.20	101.15	27	28.13	101.15	84
SUPERBLUE3	34.39	100.56	45	33.30	101.06	29	34.59	101.06	92
SUPERBLUE4	22.69	101.32	143	22.40	100.96	21	23.05	100.96	65
SUPERBLUE5	34.86	100.38	180	35.10	100.70	18	35.56	100.70	86
SUPERBLUE7	41.37	100.71	250	40.70	100.82	31	39.82	100.82	166
SUPERBLUE10	61.11	101.91	439	62.10	101.11	49	61.67	101.11	153
SUPERBLUE16	28.33	101.55	100	27.20	101.30	17	27.94	101.30	63
SUPERBLUE18	17.09	103.15	77	16.90	101.47	21	16.36	101.47	55
Avg.	+9.60%	1.003×	2.63×	+8.50%	1.000×	0.50×	+9.06%	1.007×	1.81×

Circuits	Ripple 2.0 [43]			RePlAce-r			RePlAce-r alt		
	sHPWL	RC	CPU	sHPWL	RC	CPU	sHPWL	RC	CPU
SUPERBLUE1	28.48	100.74	51	**25.89**	100.43	43	27.84	102.67	46
SUPERBLUE3	34.07	100.22	64	**30.78**	100.85	52	30.91	100.84	45
SUPERBLUE4	22.51	100.30	32	**20.94**	100.52	35	20.99	100.53	36
SUPERBLUE5	35.38	100.41	70	**33.37**	100.93	64	33.40	100.82	66
SUPERBLUE7	40.76	100.79	100	**37.10**	100.76	44	37.42	100.84	48
SUPERBLUE10	60.44	100.57	90	**58.41**	101.32	189	58.79	101.53	191
SUPERBLUE16	27.95	100.71	46	**25.46**	101.35	45	25.64	101.34	48
SUPERBLUE18	17.07	100.78	35	**14.60**	102.10	36	14.70	102.16	38
Avg.	+9.29%	0.995×	1.15×	+0.00%	1.000×	1.00×	+1.41%	1.003×	1.04×

3.6.4 State of Academic vs. Industry Placement

Finally, to help assess remaining gaps between our work and "the real world," and to demonstrate tractability of such assessment, we apply RePlAce to standard-cell placement using a foundry 28LP 8-track cell library after applying format conversion scripts as in [59]. We place-and-route four design blocks (*JPEG*, *VGA*, *LEON3MP* and *NETCARD* from [99] [162]), with up to 300K instances.[18] In our experiments, RePlAce achieves 2.4% reduction of routed wirelength on average with similar number ($<$100) of DRVs, and consumes less than 2$\times$ runtime compared to the commercial place-and-route tool. We believe that these results show encouraging progress toward bridging remaining gaps between academic and real-world placement.

3.7 Conclusion

We have described RePlAce, a single engine for global placement that, without testcase-specific tuning, achieves significant improvements over best known HPWL results on standard-cell and mixed-size benchmarks suites, as well as improvements over best known sHPWL on global routability-driven placement benchmark suites. We propose a new density function that comprehends local over-demand for area resources, leading to constraint-oriented local smoothing at a per-bin granularity. Our dynamic step size adaptation determines step size and allocates optimization effort to significantly improve solution quality without undue runtime impact. We achieve an average HPWL reduction of 2.00% over best known ISPD-2005 and ISPD-2006 benchmark results, and of 2.73% over best known MMS benchmark results. For routability-driven placement, we achieve better sHPWL on all testcases from the DAC-2012 and ICCAD-2012 benchmark suites, with on average 8.50% to 9.59% scaled HPWL reduction compared to the leading previous academic placers. To our knowledge, RePlAce is the first work to achieve overall superior solution quality across the ISPD-2005, ISPD-2006, MMS, DAC-2012 and ICCAD-2012

[18] We push placement utilization up to the limit of a 2016 release of a leading commercial tool, i.e., until post-detailed routing design rule violations (DRVs) appear without exceeding 100 DRVs in the commercial tool. The number of post-route DRVs for the commercial tool (C), and RePlAce (R) are (C, R) = (5, 3), (40, 12), (79, 68), (34, 81) for *JPEG*, *VGA*, *LEON3MP* and *NETCARD*, respectively. Due to license restrictions, we are unable to more specifically identify the commercial tool.

benchmarks with a single global placement engine. We leave to our future work several enhancements: (i) timing-driven global placement; (ii) parallel and/or distributed implementation for runtime improvement; (iii) a more systematic approach for dynamic step size adaptation with reduced runtime; (iv) integration of constraint-driven local smoothing with routability-driven placement; and (v) techniques to accelerate convergence.

3.8 Acknowledgments

Chapter 3 contains the draft submitted to *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Chung-Kuan Cheng, Andrew B. Kahng, Ilgweon Kang and Lutong Wang, "Advancing Solution Quality and Routability Validation in Global Placement", 2017. The dissertation author is the primary author of the submitted draft.

I would like to thank my coauthors Chung-Kuan Cheng, Andrew B. Kahng and Lutong Wang. I also would like to thank the authors of [82] for providing access to, and useful discussions of, their implementation. I would like to thank the authors of NTUplace3 [19] and NTUplace4h [47] for providing the executables that we use as our detailed placer.

Chapter 4

Fast and Precise Routability Analysis with Conditional Design Rules

As pin accessibility encounters more challenges due to the less number of tracks, higher pin density, and more complex design rules, routability has become one bottleneck of sub-10*nm* designs. Thus, we need a new design methodology for fast turnaround in analyzing the feasibility of the layout architecture, e.g., design rules and patterns of pin assignment. In this work, we propose a novel framework that efficiently identifies the design rule-correct routability by creating well-organized formulation. We start with a new SAT-friendly ILP formulation which satisfies conditional design rules. In the ILP-to-SAT conversion stage, we reduce the complexity of the SAT problem by utilizing a logic minimizer and further refining the SAT formula. We demonstrate that our framework performs the routability analysis within 0.24% of ILP runtime on average, while guaranteeing the precise assessment of the routability.

4.1 Introduction

Design of the routable and manufacturable layout for integrated circuits (ICs) has been more and more challenging as technology nodes are continuously advanced to sub-

10*nm* [159]. One of the major difficulties is caused by the resolution limitations of optical lithography, using 193*nm* (i.e., 193*i*) wavelength [40] [110]. Previous efforts [7] [83] [86] to improve multi-patterning techniques such as LELE (litho-etch-litho-etch), SADP and SAQP (self-aligned double and quadruple patterning) enable successful development of 10*nm* and sub-10*nm* technology nodes for foundries. Although recent multi-patterning provides advanced technology nodes, they also induce complex conditional design rules (e.g., unidirectional routing tracks per metal layer, minimum area rules, end-of-line (EOL) spacing constraints, adjacent via placement restrictions, etc.) for manufacturability which introduce new hurdles for IC physical design (PD) practitioners.

Securing pin accessibility of IC components has become a critical bottleneck during detailed routing, due to less number of routing tracks and increasing pin density [110] [136]. FinFET device with smaller pin geometry makes the pin accessibility problem even harder [1]. The number of routing tracks in a single placement row has been reduced from 12 tracks to nine/eight/seven tracks [64], and even five-track cell library with one fin is recently announced [8]. In the meanwhile, scaling metal pitch is lagging behind scaling device pitch, causing severe complications of interconnection induced by decrease of the valid access points for pins [151]. Consequently, the detailed routing step easily takes days of turnaround time (TAT), but a "successful" routing nonetheless is not guaranteed. Moreover, the growing mismatch between global-route congestion map and detailed-route design rule check (DRC) violations may endanger the on-time tapeout by demanding too many manual layout revisions [17]. To address the problem, several ILP-based (integer linear programming-based) approaches are proposed to obtain design rule-correct layout. Despite their precise formulation and ILP's optimal solutions, applying these tools to practical design procedure is difficult due to the tedious ILP optimization. The absence of fast and precise routability analysis tool at the early physical design stage (e.g., before routing) exposes the entire IC design project to high-risk unpredictability.[19] Thus, we need a new design tool with fast TAT in analyzing the feasibility of the given layout architecture, e.g., design rules and patterns of pin assignment.

[19] As an attempt to reduce the unpredictability between "routable" cell layout and physical design, cell library designers often develop a subset of cells that are essential to synthesize a small IC block. PD designers perform place-and-route (P&R), then feedback the routability assessment. However, this still takes significant time.

In this work, we propose a novel framework that efficiently identifies the design rule-correct routability for switchboxes from the placed layout through well-organized ILP and SAT (Boolean satisfiability) formulation. Our contributions are as follows.

- We propose a fast and precise routability analysis framework by using a novel SAT formulation comprehending the net structure (i.e., source-sink connectivity) and the conditional design rules. Our proposed framework produces *design rule-correct routability assessment*, offering an early "go/no-go" decision opportunity for the remaining PD procedure.

- We develop a new *SAT-friendly ILP* formulation based on multi-commodity flow. In contrast to previous ILP-based detailed routers, our ILP formulation refines all variables at the solution space as 0-1 indicator (so-called *SAT-friendly*) and defines source-sink connectivity of each net at a per-commodity granularity, to facilitate the SAT conversion.

- We devise an *ILP-to-SAT conversion* mechanism that exploits CNF (conjunctive normal form) formula by a logic minimizer [152] upon dissecting the ILP formulation. This reduces the number of literals and clauses for less complexity to further enhance runtime of the SAT solver [163].

- We demonstrate that our resulting framework gives the routability analysis *within 0.24% of ILP runtime on average*, while guaranteeing the precise assessment of the design rule-correct routability.

The rest of this chapter is organized as follows. Section 4.2 introduces the related works and our motivation. Section 4.3 describes our SAT-friendly ILP formulation using 0-1 indicators that comprehend complex design rules, which facilitates the SAT conversion. Section 4.4 presents our ILP-to-SAT conversion to reduce the problem complexity, exploited by a two-level logic minimizer. Section 4.5 discusses our experimental setups and results. Section 4.6 concludes this chapter.

4.2 Related Works and Our Approach

Cell library designers must consider BEOL-aware pin-accessibility optimization to ensure routable configurations for PD procedure [26] [135]. Also PD engineers must co-optimize both pin accessibility and pin density, while achieving design rule-correct layout as well as classical design optimization goals such as area and wirelength [48].

Recently, pin-accessibility optimization under DFM (design for manufacturability) constraints have been massively studied as shown in Figure 4.1. Pin accessibility- and BEOL-aware cell layout optimizations [26] [110] [135], and three-dimensional (3D) monolithic standard cells to improve pin accessibility [112] are introduced. Previous ISPD-2014 [146] and ISPD-2015 [11] contests are dedicated to the detailed routing-driven placement. Placement migrations to mitigate local routing congestion [117] [125] and pin accessibility-aware detailed placement refinements [28] [29] are proposed. For detailed

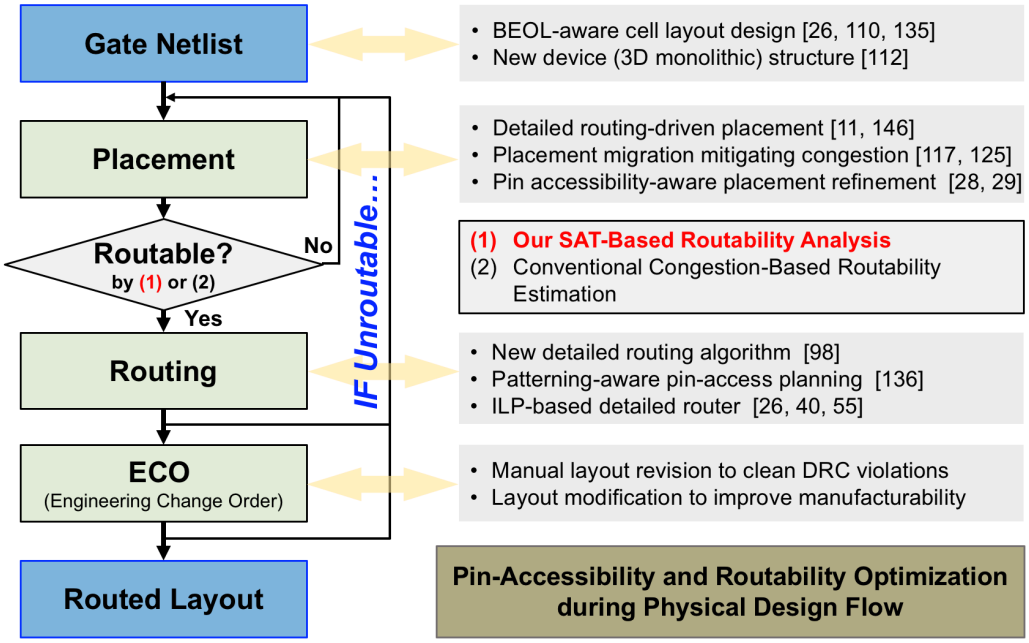


Figure 4.1: Efforts to secure the pin-accessibility during the PD procedure. Failure to produce routable (or routed) design in each step indicates loop-back of PD procedure, causing additional design cost. Our SAT-based routability analysis (in red font) enables a fast and precise assessment.

routing, an algorithm for dense pin clusters [98] and a pin-access planning framework comprehending SADP [136] are presented. Despite the previous studies above, there still exist high risks of design failure for achieving proper pin accessibility during the routing step, causing undesired additional design cost.

By virtue of its ability to obtain optimal solutions, ILP-based optimization has been widely applied to global and detailed routers, on top of the multi-commodity flow theory. Recent works in [26] [40] [55] describe ILP-based detailed routing formulation for standard cell design [26] and detailed routers [40] [55], comprehending conditional design rules for advanced technologies. By nature of ILP seeking optimal solutions, they are nevertheless timely-expensive and unaffordable for quick routability analysis.

Due to the high complexity in expressing "countable" design metrics and constraints into CNF representation, adopting SAT for PD has been limited to standard cell routing [106], escape routing for board design [84], and special-purpose ICs such as FPGA [32] [90] [94] and cross-referencing biochips [143]. In [106], the authors use a maze routing algorithm to find all possible routes within a bounding box, and create a Boolean literal for each route.

Conventional routability predictions ((2) in Figure 4.1) after placement are based on congestion map estimation, often producing false-positive or -negative analyses that induce unnecessary design efforts. Therefore, we need a new design tool for fast and precise routability analysis for the given layout architecture, e.g., design rules and patterns of pin assignment. In the remaining sections, we describe our novel SAT-based routability analysis ((1) in Figure 4.1) on top of SAT-friendly ILP formulation, which enables design rule-correct routability assessment.

4.3 SAT-Friendly ILP Formulation

As discussed above, to adopt Boolean satisfiability (SAT) for PD is nontrivial. Countable design metrics and constraints can be dealt with integer linear programming (ILP), but careful formulation is required to preserve the Boolean nature of 0-1 variables [2]. In this section, we describe a new ILP-based routing formulation that refines all variables at the solution space as 0-1 indicator and defines source-sink connectivity of

each net at a per-commodity granularity[20] (i.e., per each sink), to facilitate our ILP-to-SAT conversion (so-called *SAT-friendly ILP*). Our ILP formulation captures discrepancies between minimum-cost wirelength and design rule-correct metal segments and optimizes both. Table 4.1 presents the notations.

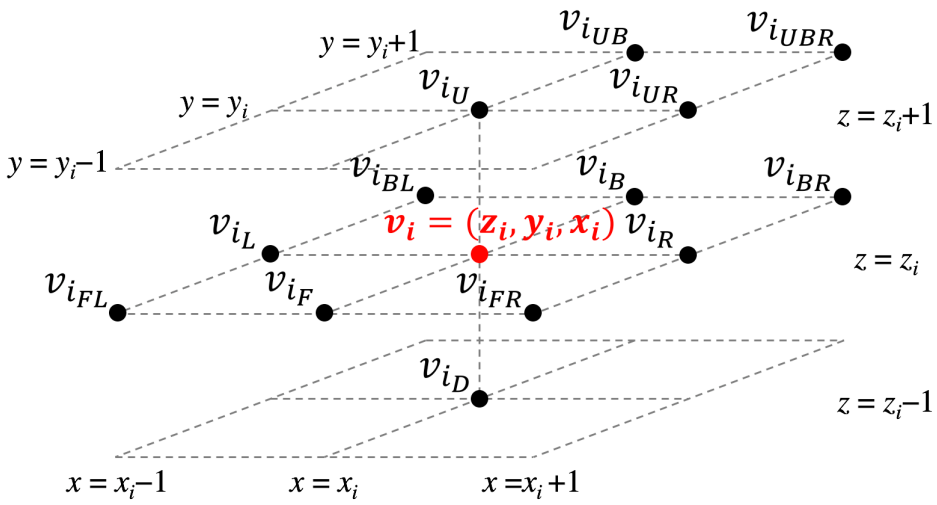


Figure 4.2: Adjacent vertices (for our ILP formulation) of v_i in the routing graph G.

4.3.1 ILP-Based Detailed Routing Optimization

We formulate our ILP-based routing formulation on top of the multi-commodity flow network theory. In the three-dimensional (3D) routing graph $G = (V, A)$, we represent circuit components (e.g., sources, sinks) and available routing resources (e.g., horizontal and vertical tracks on multiple metal layers, inter-layer vias). Each vertex v_i is mapped to coordinate (z_i, y_i, x_i), where z_i, y_i, and x_i are metal layer, horizontal and vertical routing tracks, respectively. We define a directed edge $a_{i,j}$ and an undirected edge $e_{i,j}$, including inter-layer vias. We refine all the variables and constants to have either 0 or 1 to facilitate ILP-to-SAT conversion.[22]

[20] As a result, all constants in our ILP formulation have either 0 or 1.

[21] The symbol d is L (Left), R (Right), F (Front), B (Back), U (Up), D (Down), or a combination of these directions, e.g., FL (FrontLeft), FR (FrontRight), BL (BackLeft), BR (BackRight), UR (UpperRight), UB (UpperBack), and UBR (UpperBackRight) as shown in Figure 4.2.

[22] The only exception is made to define $f_m^k(i) = -1$ when $v_i = t_m^k$. When we perform the ILP-to-SAT conversion, we associate -1 with 'True', and we check the other variables in the ILP formula to determine the feasibility of the ILP's (in)equality.

Table 4.1: Notations for ILP and SAT formulation.

Term	Description
$G(V,A)$	Three-dimensional (3D) routing graph
V	Set of vertices in the routing graph G
v_i	A vertex with the coordinate (z_i, y_i, x_i)
A	Set of directed edges in the routing graph G
$a_{i,j}$	A directed edge from v_i to v_j
E	Set of undirected edges in the routing graph G
$e_{i,j}$	An undirected edge between v_i and v_j
$c_{m_{i,j}}$	Cost for metal segment on $e_{i,j}$
$c_{w_{i,j}}$	Cost for wire segment on $e_{i,j}$
N	Set of multi-pin nets in the given routing box
n^k	k^{th} multi-pin net
s^k	A source of n^k
T^k	Set of sinks in n^k
t_m^k	m^{th} sink of n^k
f_m^k	m^{th} commodity flow of n^k heading to t_m^k
$a_{i,j}^k$	0-1 indicator if $a_{i,j}$ is used for n^k
$e_{i,j}^k$	0-1 indicator if $e_{i,j}$ is used for n^k
$f_m^k(i)$	Flow variable on v_i for commodity t_m^k
$f_m^k(i,j)$	0-1 indicator if $a_{i,j}$ is used for commodity t_m^k
$u_m^k(i,j)$	0-1 indicator if $e_{i,j}$ is used for commodity t_m^k
$m_{i,j}$	0-1 indicator if there is a metal segment on $e_{i,j}$
$w_{i,j}$	0-1 indicator if there is a wire segment on $e_{i,j}$
v_{i_d}	A d-directional[21]adjacent vertex of v_i
$g_{d,i}$	0-1 indicator if v_i forms d-side EOL of a metal segment

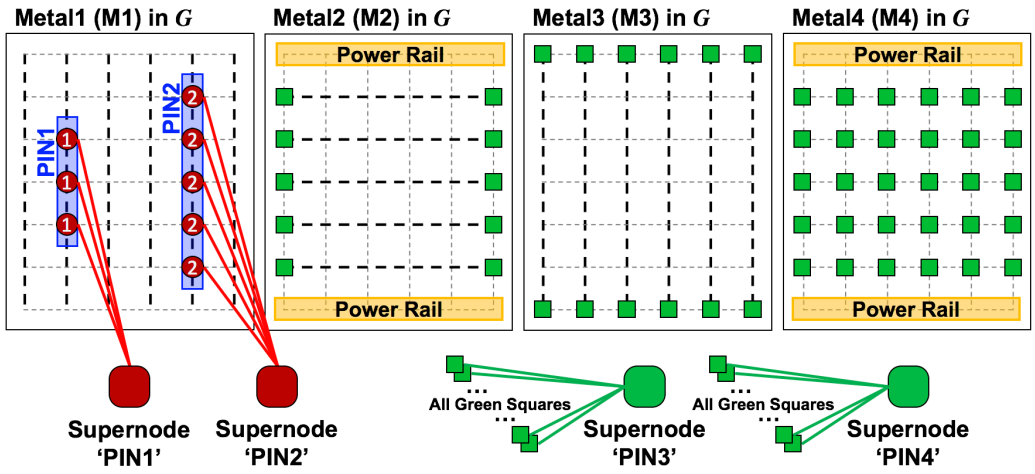


Figure 4.3: An example of supernodes. $PIN1$ and $PIN2$ respectively cover three and five vertices on $M1$ layer. Outer pins ($PIN3$ and $PIN4$) are connected to boundary vertices of G.

Objective. We minimize both the weighted total wirelength and the weighted total metal length. We define inter-layer via cost as 4 and wire/metal cost on the same metal layer as 1. The wire segments are not necessarily the same with the metal segments since we often assign extra metal segments to avoid design rule check (DRC) violations such as minimum area rule (MAR), end-of-line (EOL) spacing, etc (as shown in Figure 4.10). Our objective function is shown in Equation (4.1).

$$\textbf{Minimize}: \quad Cost = \sum_{e_{i,j} \in E} \left(c_{w_{i,j}} w_{i,j} + c_{m_{i,j}} m_{i,j} \right) \tag{4.1}$$

Supernode. A commodity must outflow from a vertex (i.e., source) and inflow to another vertex (i.e., sink). Since a pin includes multiple vertices, we generate supernodes for each pin, which cover vertices on each pin. Figure 4.3 illustrates supernodes in G. A supernode for a pin on $M1$ is connected to vertices covering the pin (red circles on $PIN1$ and $PIN2$). A supernode for a pin outside of G is connected to vertices at the boundary of G (i.e., left/right/front/back-most vertices on $M2$ and $M3$, and all accessible vertices on $M4$) as depicted in green squares. We avoid placing horizontal metal segments and inter-layer vias in the yellow boxes, reserved for power rail.

Unidirectional Routing. We assume unidirectional routing. We simply do not generate edges that are not in the preferred direction.

Commodity Flow Conservation (CFC). The CFC on each vertex and its connected edges ensures source-to-sink connectivity of each commodity flow, as shown in Constraint (4.2).

$$f_m^k(i) + \sum_{v_i : a_{j,i} \in A} f_m^k(j,i) - \sum_{v_i : a_{i,j} \in A} f_m^k(i,j) = 0, \quad \forall v_i \in V,\ \forall n^k \in N,\ \forall t_m^k \in T^k \tag{4.2}$$

$$f_m^k(i) = \begin{cases} 1, & \text{if } v_i = s^k \\ -1, & \text{else if } v_i = t_m^k \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

Based on the given layout definition, Constraint (4.3) defines the commodity flow variables on each vertex at a per-net and per-commodity granularity. If a vertex v_i is s^k, $f_m^k(i) = 1$. Since a net has a single source, $f_m^k(i) = 1$ for all commodities belonging n^k. If a vertex v_i is t_m^k, $f_m^k(i) = -1$ (see Footnote 22). For any vertex v_i that is neither source nor sink, $f_m^k(i) = 0$. In Constraint (4.2), the second and third terms are the sum of commodity flows inflowing to v_i and outflowing from v_i, respectively. If $v_i = s^k$ (or t_m^k), the second and third terms must be 0 and 1 (or 1 and 0). Otherwise, both terms are 1 (or 0) when v_i is (or is not) used for commodity flow f_m^k.

Edge Assignment (EA). To obtain Steiner tree of net n^k, we determine $a_{i,j}^k$ by overlapping each commodity flow. Note that our ILP is formulated at a per-commodity granularity of each net. As shown in Constraint (4.4), $a_{i,j}^k = 1$ when $f_m^k(i,j) = 1$. If $f_m^k(i,j) = 0$, $a_{i,j}^k$ can be either 0 or 1, ensuring multi-commodity flow of n^k.

$$a_{i,j}^k - f_m^k(i,j) \geq 0, \quad \forall a_{i,j} \in A,\ \forall n^k \in N,\ \forall t_m^k \in T^k \tag{4.4}$$

Exclusiveness Use of Vertex (EUV). A vertex v_i should be used by only one net. Constraint (4.5) ensures that there are no intersecting nets on any vertex. To preserve multi-pin nets, we only constrain inflows to v_i, thus v_i can have multiple outflows if required. When $v_i = s^k$, the sum of inflows to v_i is 0. Otherwise, the sum is 1.

$$\sum_{v_i : n^k \in N} a^k_{j,i} \leq \begin{cases} 0, & \text{if } v_i = s^k \\ 1, & \text{otherwise} \end{cases}, \quad \forall v_i \in V \tag{4.5}$$

Metal Segment (and Exclusiveness Use of Edge) (MS). We adopt $m_{i,j}$ to determine whether a metal segment is on $e_{i,j}$ or not. Constraint (4.6) also ensures that there are no intersecting nets on any edge. The RHS term is the total sum of $a^k_{i,j}$ and $a^k_{j,i}$ for all $n^k \in N$. The sum is at most 1, ensuring the exclusiveness use of $e_{i,j}$.

$$m_{i,j} = \sum_{n^k \in N} \left(a^k_{i,j} + a^k_{j,i}\right), \quad \forall v_i \in V \tag{4.6}$$

Wire Segment (WS). We determine $w_{i,j}$ by logically ORing all commodity flows on $e_{i,j}$ for all $n^k \in N$ as shown in Constraint (4.7).

$$w_{i,j} = \bigvee_{n^k \in N} \bigvee_{t^k_m \in T^k} \left(f^k_m(i,j) \vee f^k_m(j,i)\right), \quad \forall v_i \in V \tag{4.7}$$

Inspired by [105], we transform Constraint (4.7) into a set of linear expressions as shown in Constraint (4.8). By combined, Constraint (4.8) determines $w_{i,j} = 1$ if there are commodity flows on $e_{i,j}$.

$$\begin{aligned} w_{i,j} &\leq \sum_{n^k \in N} \sum_{t^k_m \in T^k} \left(f^k_m(i,j) + f^k_m(j,i)\right) \quad ; \\ w_{i,j} &\geq f^1_1(i,j) \quad ; \quad w_{i,j} \geq f^1_1(j,i) \quad ; \\ &\cdots \quad ; \quad \cdots \quad ; \\ w_{i,j} &\geq f^k_m(i,j) \quad ; \quad w_{i,j} \geq f^k_m(j,i) \end{aligned} \tag{4.8}$$

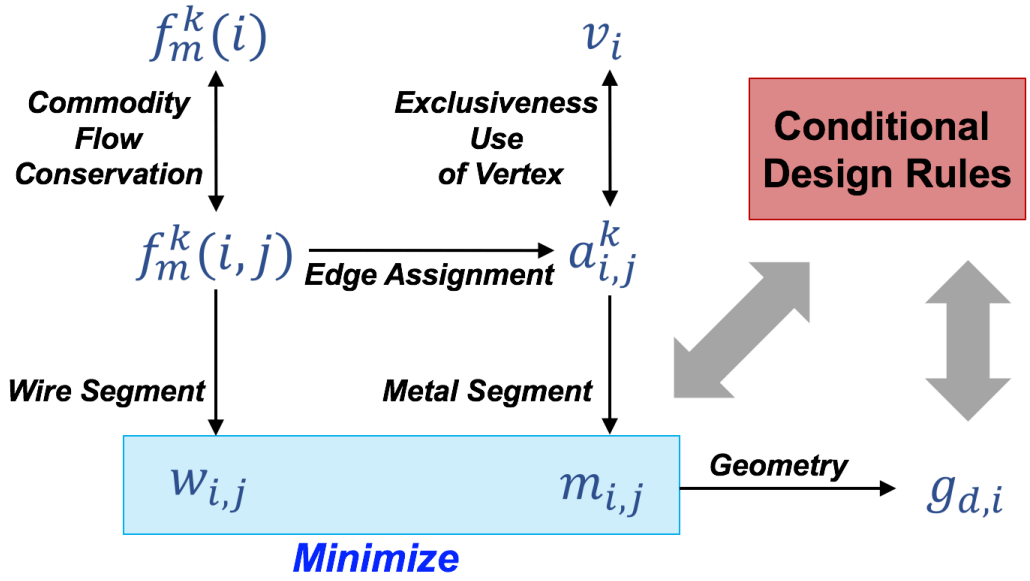


Figure 4.4: Relations of our ILP-based routing optimization formulas and variables.

Figure 4.4 illustrates relations of our ILP-based routing optimization formulas and variables. The geometric variable $g_{d,i}$ and conditional design rules will be formulated in Section 4.3.2.

4.3.2 ILP-Based Design Rule Formulation

In this section, we describe our ILP-based conditional design rule formulation.

Geometric Variable (GV). Geometric variable $g_{d,i}$ determines if v_i forms a d-directional end of line (EOL) for a metal segment where d is an adjacent vertex of v_i. Geometric variables are logically expressed as shown in Constraint (4.9).

$$\begin{aligned} g_{L,i} &= \neg m_{i_L,i} \wedge m_{i,i_R}, \quad \forall v_i \in V \\ g_{R,i} &= m_{i_L,i} \wedge \neg m_{i,i_R}, \quad \forall v_i \in V \\ g_{F,i} &= \neg m_{i_F,i} \wedge m_{i,i_B}, \quad \forall v_i \in V \\ g_{B,i} &= m_{i_F,i} \wedge \neg m_{i,i_B}, \quad \forall v_i \in V \end{aligned} \tag{4.9}$$

Figure 4.5 shows an example to determine $g_{d,i}$. Since the left- and right-directional EOLs (depicted in red and blue bars) are respectively located at v_2 and v_4, $g_{L,2} = 1$ and $g_{R,4} = 1$. We convert logical Constraint (4.9) into linear Constraint (4.10) since $x \wedge y = z \Leftrightarrow z \leq x$, $z \leq y$, $z \geq x + y - 1$ and $\neg x \Leftrightarrow 1 - x$, as described in [40].

$$\begin{aligned} &g_{L,i} \leq 1 - m_{i_L,i} \;\; ; \;\; g_{L,i} \leq m_{i,i_R} \;\; ; \;\; g_{L,i} \geq m_{i,i_R} - m_{i_L,i} \\ &g_{R,i} \leq m_{i_L,i} \;\; ; \;\; g_{R,i} \leq 1 - m_{i,i_R} \;\; ; \;\; g_{R,i} \geq m_{i_L,i} - m_{i,i_R} \\ &g_{F,i} \leq 1 - m_{i_F,i} \;\; ; \;\; g_{F,i} \leq m_{i,i_B} \;\; ; \;\; g_{F,i} \geq m_{i,i_B} - m_{i_F,i} \\ &g_{B,i} \leq m_{i_F,i} \;\; ; \;\; g_{B,i} \leq 1 - m_{i,i_B} \;\; ; \;\; g_{B,i} \geq m_{i_F,i} - m_{i,i_B} \end{aligned} \tag{4.10}$$

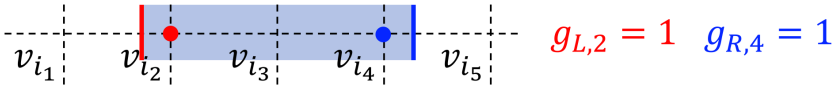


Figure 4.5: An example to determine $g_{d,i}$.

Minimum Area Rule (MAR). Each disjoint metal segment should be larger than the minimum manufacturable size. Constraint (4.11) ensures that there are no metal segments violating the MAR. In our formulation, we assume that a metal segment must cover at least three vertices. Figure 4.6 illustrates the MAR formula.

$$g_{L,i} + g_{R,i} + g_{L,i_R} + g_{R,i_R} \leq 1, \quad g_{F,i} + g_{B,i} + g_{F,i_B} + g_{B,i_B} \leq 1, \quad \forall v_i \in V \tag{4.11}$$

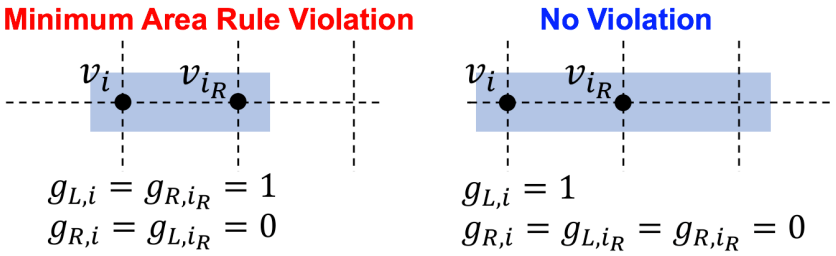


Figure 4.6: An example of the minimum area rule.

End-of-Line (EOL) Spacing Rule. The distance between each EOL (i.e., "tip") of two metal segments that are coming from opposite directions should be greater than the minimum spacing distance (tip-to-tip spacing rule). In our formulation, we assume that the

minimum distance between any of two opposite tips should be larger than L_1 norm (i.e., Manhattan distance) of two vertices in G, as described in Constraint (4.12) and Figure 4.7. A notation i_{LL} (or i_{RR}) denotes left-adjacent (or right-adjacent) vertex of v_{i_L} (or v_{i_R}).

$$\begin{aligned}
&g_{L,i} + g_{R,i_{FL}} \leq 1 \quad ; \quad g_{L,i} + g_{R,i_L} + g_{R,i_{L_L}} \leq 1 \quad ; \quad g_{L,i} + g_{R,i_{BL}} \leq 1, \quad \forall v_i \in V \\
&g_{R,i} + g_{L,i_{FR}} \leq 1 \quad ; \quad g_{R,i} + g_{L,i_R} + g_{L,i_{R_R}} \leq 1 \quad ; \quad g_{R,i} + g_{L,i_{BR}} \leq 1, \quad \forall v_i \in V \\
&g_{F,i} + g_{B,i_{FL}} \leq 1 \quad ; \quad g_{F,i} + g_{B,i_F} + g_{B,i_{F_F}} \leq 1 \quad ; \quad g_{F,i} + g_{B,i_{FR}} \leq 1, \quad \forall v_i \in V \\
&g_{B,i} + g_{F,i_{BL}} \leq 1 \quad ; \quad g_{B,i} + g_{F,i_B} + g_{F,i_{B_B}} \leq 1 \quad ; \quad g_{B,i} + g_{F,i_{BR}} \leq 1, \quad \forall v_i \in V
\end{aligned} \tag{4.12}$$

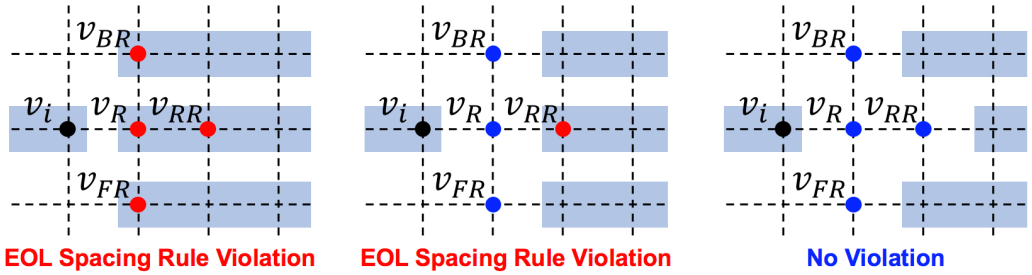


Figure 4.7: An example of the end-of-line (EOL) spacing rule.

Via Rules (VR). Inter-layer via placements in side-by-side as well as in diagonal direction are strictly restricted as described in Constraint (4.13) (i.e., via-to-via spacing rule). Also inter-layer via placements on the top of another inter-layer via are not allowed as shown in Constraint (4.14) (i.e., stacked-via placement restriction).

$$m_{i,i_U} + m_{i_R,i_{UR}} + m_{i_B,i_{UB}} + m_{i_{BR},i_{UBR}} \leq 1, \quad \forall v_i \in V \tag{4.13}$$

$$m_{i_D,i} + m_{i,i_U} \leq 1, \quad \forall v_i \in V \tag{4.14}$$

Power Rail. Since the top- and bottom-most routing tracks for each placement row are reserved for power rails, we prohibit placing horizontal metal segments and inter-layer vias on those power rails. We trivially set all 0-1 indicators on corresponding edges as 0.

4.4 ILP-to-SAT Conversion

We describe our ILP-to-SAT conversion that considers to improve the runtime of SAT solvers [163]. A general SAT solver takes input in *conjunctive normal form (CNF)*: an AND of ORs of *literals*, i.e., propositional variables and their negations. A CNF formula is a conjunction (AND) of *clauses* that are expressed in a disjunction (OR) of literals. We reduce our SAT's complexity by utilizing the logic minimizer [152] upon dissecting the structures and patterns of our ILP formulas. Moreover, we develop a *reduced SAT formulation*, which further decreases the numbers of literals and clauses in our CNF formula by replacing directed edges into undirected edges.

4.4.1 Advantage of Our SAT-Friendly ILP

To facilitate the ILP-to-SAT conversion, we devise two techniques for our ILP formulation: (i) refining all the variables as 0-1 indicator and (ii) formulating source-sink connectivity of each net at a per-commodity granularity. To convert an ILP constraint into a set of clauses in a CNF formula, we analyze feasibilities for all combinations of values of each variable within the ILP constraint. Each feasible combination corresponds to a single clause. This would present a huge chance of having exponential numbers of clauses. For example, suppose that an ILP constraint is expressed as:

$$x_1 + x_2 + \cdots + x_n \leq y, \tag{4.15}$$

where $x_i, y \in Z^*$, Z^* is a set of nonnegative integers. For nonnegative x_i and y, the number of feasible cases that satisfy Constraint (4.15) is $\binom{n(y+1)}{y}$, which yields $\binom{n(y+1)}{y}$ of y-literal clauses. For example, when $n = 10$ and $y = 10$, the number of clauses we should consider is larger than $1.0e^{18}$, inapplicable to the ILP-to-SAT conversion. Thus, we refine all the variables in our ILP as 0-1 indicator to reduce the complexity. When we restrict x_i to 0-1 indicator, Constraint (4.15) yields $\binom{2n}{y}$ y-literal clauses, approximately $1.8e^5$ clauses when $n = 10$ and $y = 10$. However, this is still insufficient for practical ILP-to-SAT conversion since an ILP problem has many constraints (ranging $10K$-$2.5M$ in our work) with many variables (up to thousands). To further facilitate our ILP-to-SAT conversion, we formulate the source-sink connectivity at a per-commodity (i.e., per-sink) granularity,

inducing RHS constant (i.e., y in Constraint (4.15)) to be 0 or 1. By restricting both x_i and y to 0-1 valued variables and constants, Constraint (4.15) only yields $n+2$ clauses, i.e., 12 clauses when $n = 10$ and $y = 10$. This reduction enables the efficient ILP-to-SAT conversion.

4.4.2 Logic Simplification

We exploit our SAT formulation by the two-level logic minimizer *Espresso* [152] to obtain the minimal CNF representations providing the functionally-equivalent CNF formula with the fewest number of clauses and literals. By virtue of regularity of the 3D routing graph G, ILP constraints derived by the same ILP formula have the similar structure. Also sets of clauses oriented from the same ILP formula share common patterns of CNF representation. With this in mind, we dissect the structures of ILP constraints and their patterns of the minimal CNF representations. We utilize the preprocessed *Espresso*'s results as the predetermined minimal CNF representations for each ILP formula. As a result, we generate the minimal CNF formula without calling *Espresso* during our ILP-to-SAT conversion.

Although a CNF formula is an AND of ORs, i.e., a product of sums, we feed the ON-set (i.e., a set of all feasible input combinations of the ILP constraint) in sum-of-product form to *Espresso*. *Espresso* swaps the ON-set and OFF-set to simplify the OFF-set, so *Espresso*'s solution is a set of all infeasible cases for the given ILP constraint. To obtain minimal CNF representations in product-of-sum form, we complement *Espresso*'s solution.

4.4.3 Reduced SAT Formulation

We propose a *reduced SAT formulation* to further reduce the numbers of literals and clauses in the CNF formula by replacing directed edges into undirected edges. We can ignore directions of flows since we mainly focus on source-sink connectivities in analyzing the routability. Instead of $a^k_{i,j}$ and $f^k_m(i,j)$ in ILP formulation, we associate $e^k_{i,j} = a^k_{i,j} \veebar a^k_{j,i}$ and $u^k_m(i,j) = f^k_m(i,j) \veebar f^k_m(j,i)$ with the directed-edge-related terms in the

ILP-based routing optimization (in Section 4.3.1), where $\veebar$ denotes XOR. Note that we only reduce the SAT-based routability formulation since the design-rule formulas do not have directed-edge-related terms.

Reduced SAT-Based Routability Formulation. Logical Expressions (4.16,4.18) and (4.17,4.19) represent the feasibilities of ILP formulas, CFC (Constraints (4.2-4.3)) and EUV (Constraint (4.5)), respectively. Expressions (4.16-4.17) determine the feasibilities when $v_i \neq s^k, t_m^k$.

$$\mathbf{F}_{CFC} = \left(\neg f_m^k(i) \wedge u_m^k(i,p) \wedge u_m^k(i,q) \wedge \bigwedge_{v_i:e_{i,j} \in E, j \neq p, j \neq q} \neg u_m^k(i,j) \right) \vee \left(\neg f_m^k(i) \wedge \bigwedge_{v_i:e_{i,j} \in E} \neg u_m^k(i,j) \right) \tag{4.16}$$

$$\mathbf{F}_{EUV} = \left(e_{i,p}^k \wedge \bigvee_{v_i:e_{i,j} \in E, j \neq p} e_{i,j}^k \wedge \bigwedge_{v_i:n^r \in N, r \neq k} \neg e_{i,j}^r \right) \vee \bigwedge_{v_i:n^k \in N} \neg e_{i,j}^k \tag{4.17}$$

Expressions (4.18-4.19) determine the feasibilities when $v_i = s^k,\ t_m^k$.

$$\mathbf{F}_{CFC} = f_m^k(i) \wedge u_m^k(i,p) \wedge \bigwedge_{v_i:e_{i,j} \in E, j \neq p} \neg u_m^k(i,j) \tag{4.18}$$

$$\mathbf{F}_{EUV} = e_{i,p}^k \wedge \bigwedge_{v_i:e_{i,j} \in E, j \neq p} \neg e_{i,j}^k \wedge \bigwedge_{v_i:n^r \in N, r \neq k} \neg e_{i,j}^r \tag{4.19}$$

Logical Expressions (4.20-4.22) represent the feasibilities of ILP Constraints (4.4,4.6-4.7), i.e., EA, MS, and WS, respectively.

$$\mathbf{F}_{EA} = e_{i,j}^k \vee \neg u_m^k(i,j) \tag{4.20}$$

$$\mathbf{F}_{MS} = \left(m_{i,j} \wedge e_{i,j}^k \wedge \bigwedge_{n^r \in N, r \neq k} \neg e_{i,j}^r \right) \vee \left(\neg m_{i,j} \wedge \bigwedge_{n^k \in N} \neg e_{i,j}^k \right) \tag{4.21}$$

$$\mathbf{F}_{WS} = \left(w_{i,j} \wedge \bigvee_{n^k \in N} \bigvee_{t_m^k \in T^k} u_m^k(i,j) \right) \vee \left(\neg w_{i,j} \wedge \bigwedge_{n^k \in N} \bigwedge_{t_m^k \in T^k} \neg u_m^k(i,j) \right) \tag{4.22}$$

The routing feasibility $\mathbf{F}$ is defined as

$$\mathbf{F} = \mathbf{F}_{CFC} \wedge \mathbf{F}_{EUV} \wedge \mathbf{F}_{EA} \wedge \mathbf{F}_{MS} \wedge \mathbf{F}_{WS}. \tag{4.23}$$

SAT-Based Design Rule Formulation. Logical Expression (4.24) represents the left- and right-directional EOL of a metal segment for the CNF formula (corresponding ILP Constraint (4.9)). The front- and back-directional representation is derived by switching L and R to F and B, respectively.

$$\begin{aligned}
\mathbf{D}_{GV_L} &= (g_{L,i} \wedge \neg m_{i_L,i} \wedge m_{i,i_R}) \vee (\neg g_{L,i} \wedge \neg m_{i,i_R}) \vee (\neg g_{L,i} \wedge m_{i_L,i}) \\
\mathbf{D}_{GV_R} &= (g_{R,i} \wedge m_{i_L,i} \wedge \neg m_{i,i_R}) \vee (\neg g_{R,i} \wedge \neg m_{i_L,i}) \vee (\neg g_{R,i} \wedge m_{i,i_R}) \\
\mathbf{D}_{GV} &= \mathbf{D}_{GV_L} \wedge \mathbf{D}_{GV_R} \wedge \mathbf{D}_{GV_F} \wedge \mathbf{D}_{GV_B}
\end{aligned} \tag{4.24}$$

Logical Expressions (4.25-4.28) represent ILP Constraints (4.11-4.14), i.e., MAR, EOL, and VR. For EOL, we only present $g_{L,i}$-related formulation ($\mathbf{D}_{EOL_L}$) as $\mathbf{D}_{EOL_R}$, $\mathbf{D}_{EOL_F}$, and $\mathbf{D}_{EOL_B}$ are similarly derived.

$$\begin{aligned}
\mathbf{D}_{MAR} &= (\neg g_{L,i} \wedge \neg g_{R,i} \wedge \neg g_{L,i_R}) \vee (\neg g_{L,i} \wedge \neg g_{R,i} \wedge \neg g_{R,i_R}) \\
&\quad \vee (\neg g_{L,i} \wedge \neg g_{L,i_R} \wedge \neg g_{R,i_R}) \vee (\neg g_{R,i} \wedge \neg g_{L,i_R} \wedge \neg g_{R,i_R})
\end{aligned} \tag{4.25}$$

$$\mathbf{D}_{EOL_L} = \neg g_{L,i} \vee (\neg g_{R,i_{FL}} \wedge \neg g_{R,i_L} \wedge \neg g_{R,i_{LL}} \wedge \neg g_{R,i_{BL}}) \tag{4.26}$$

$$\mathbf{D}_{EOL} = \mathbf{D}_{EOL_L} \wedge \mathbf{D}_{EOL_R} \wedge \mathbf{D}_{EOL_F} \wedge \mathbf{D}_{EOL_B} \tag{4.27}$$

$$\begin{aligned}
\mathbf{D}_{VR} &= (\neg m_{i_R,i_{UR}} \wedge \neg m_{i_BR,i_{UB}} \wedge \neg m_{i_{BR},i_{UBR}} \wedge \neg m_{i_D,i}) \vee (\neg m_{i,i_U} \wedge \neg m_{i_B,i_{UB}} \wedge \neg m_{i_{BR},i_{UBR}}) \\
&\quad \vee (\neg m_{i,i_U} \wedge \neg m_{i_R,i_{UR}} \wedge \neg m_{i_{BR},i_{UBR}}) \vee (\neg m_{i,i_U} \wedge \neg m_{i_R,i_{UR}} \wedge \neg m_{i_B,i_{UB}})
\end{aligned} \tag{4.28}$$

The design rule correctness $\mathbf{D}$ is defined as

$$\mathbf{D} = \mathbf{D}_{GV} \wedge \mathbf{D}_{MAR} \wedge \mathbf{D}_{EOL} \wedge \mathbf{D}_{VR}. \tag{4.29}$$

SAT-Based Design Rule-Correct Routability Analysis. Combined with the given input layout representation $\mathbf{I}$, the design rule-correct routability $\mathbf{R}$ is determined as shown in Expression (4.30).

$$\mathbf{R} = \mathbf{F} \wedge \mathbf{D} \wedge \mathbf{I} \tag{4.30}$$

4.5 Experiments

We now describe the overall flow of our proposed framework and the experimental results through the ILP-, the SAT- and the reduced SAT based routability analysis.

4.5.1 Overall Flow of the Proposed Framework

Figure 4.8 presents the overall flow of our framework. Red and blue rectangles respectively represent our toolchain and each step's inputs. We separately depict the logic simplification step as red-dotted rectangle at the outside of our routability analysis procedure since we perform the logic simplification only once based on the patterns and structures of ILP formulas. We refine our ILP-to-SAT conversion by exploiting the simplified CNF representations through the logic minimizer *Espresso* [152]. The blue arrow presents the ILP-based detailed routing procedure. We adopt *CPLEX 12.7.1* [157] as our ILP solver. The green arrows present the SAT-based routability analysis procedures. We employ a popular open-source multi-threading SAT solver *Plingeling 2.4.1* [163]. We implement our framework as a chain of scripts to create artificial testcases, generate the ILP formulation, and convert the ILP into the SAT and the reduced SAT formulation.

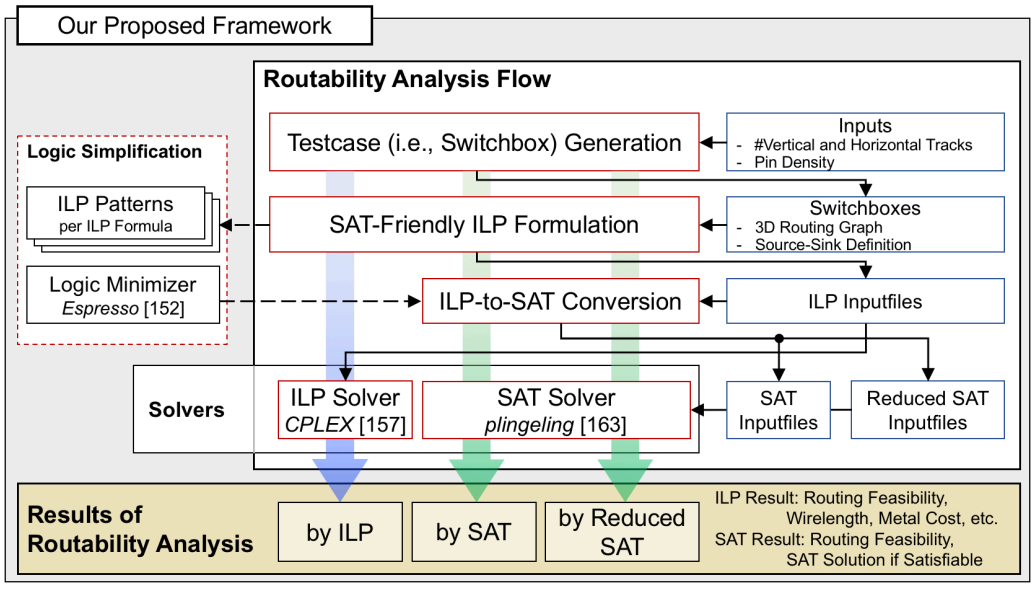


Figure 4.8: Overall flow of our routability analysis.

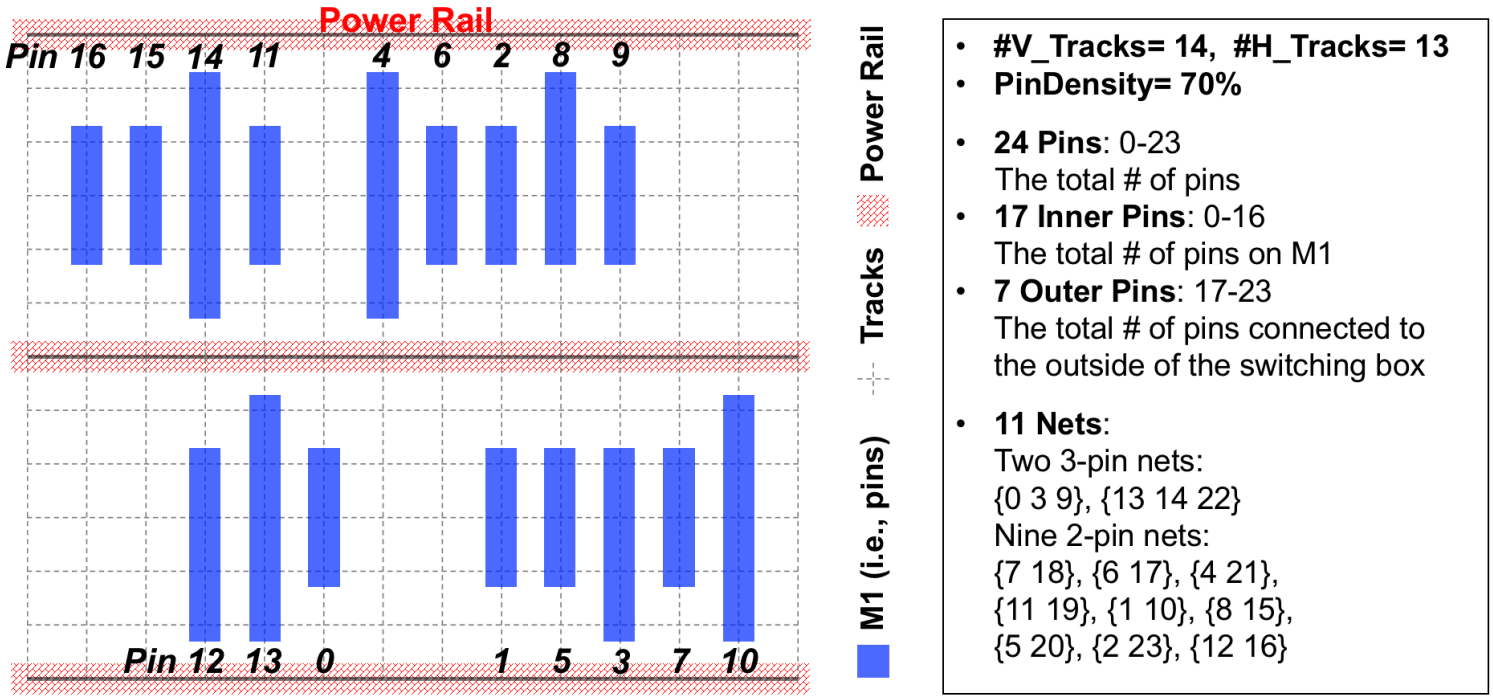


Figure 4.9: An example layout with 14×13 vertial and horizontal tracks. 70% pin density. The total #pins=24. 17 pins are on $M1$ and seven pins are outside of the layout (#outer pins are determined by Rent's rule [147]). We have 11 nets (i.e., two 3-pin nets and nine 2-pin nets).

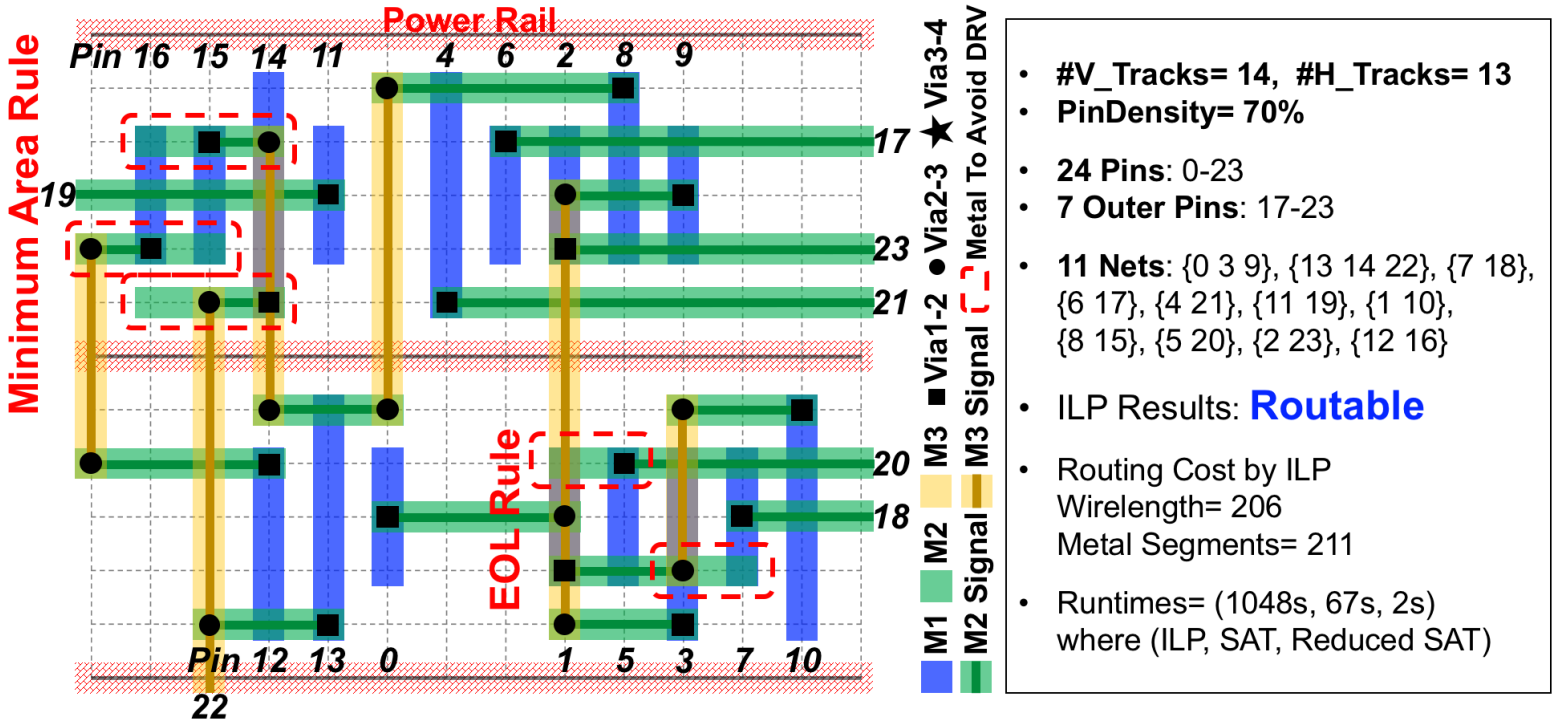


Figure 4.10: The ILP-based optimal routing solution for the layout in Figure 4.9. Cost = 417. Five more metal segments are assigned to avoid design rule check (DRC) violations (red dotted circles).

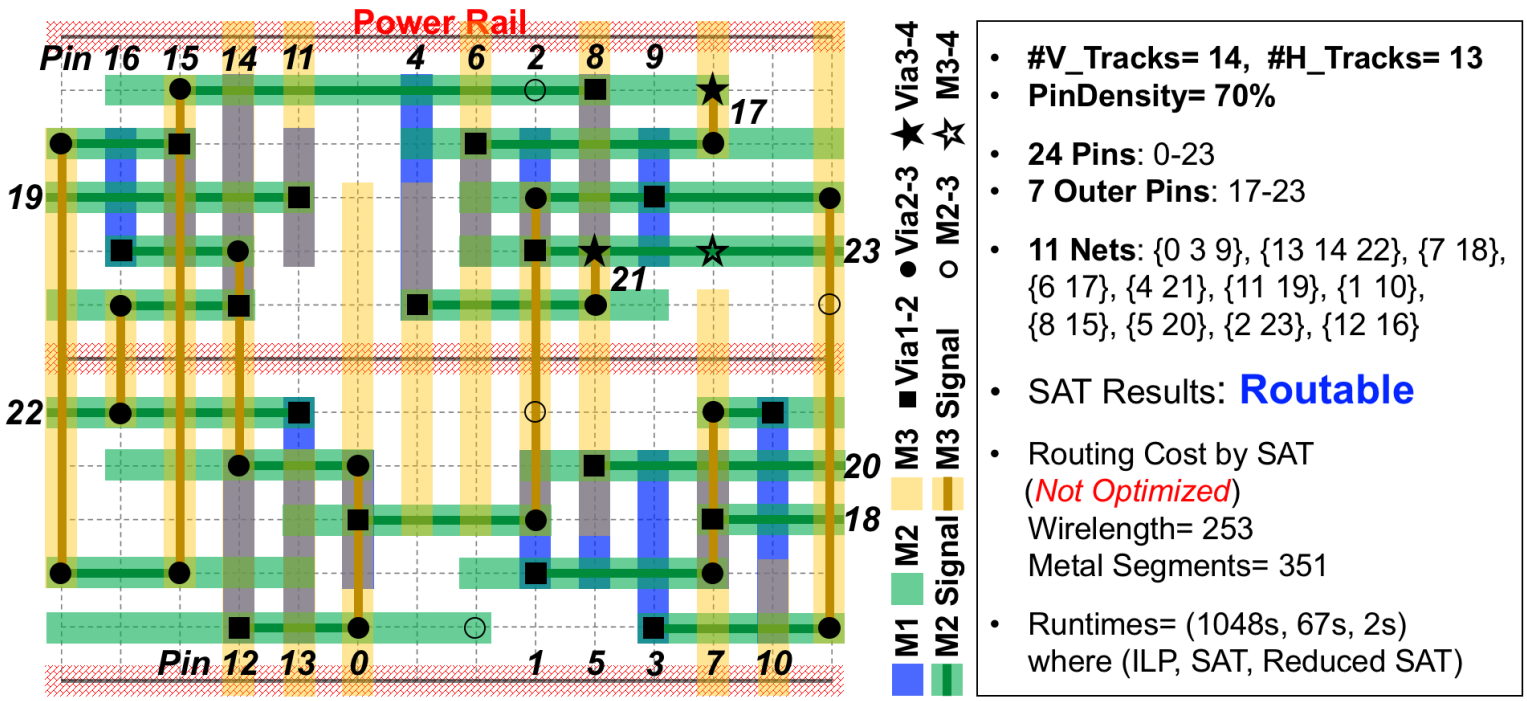


Figure 4.11: The reduced SAT-based routing solution for the layout in Figure 4.9. Cost = 604. The solution is not optimal, but takes only 0.19% of ILP's runtime, thus we can anaylze the routability quickly.

For the testcase generation, we specify the inputs as the numbers of vertical and horizontal tracks, and pin density for the switchbox. We assume 7-track cell library, i.e., five tracks for each placement row are applicable for detailed routing. Each pin on $M1$ layer cover 3-5 vertices. Based on [147], we consider *Rent's rule* for the artificial testcase generation, indicating that the numbers of outward net connections, outer pins, multi-pin nets, and multi-fanout gates in the switchbox are practical. We do not apply any detailed-placement refinement techniques (such as cell padding) for routability (i.e., pin accessibility) improvement. With the given switchbox information, we sequentially generate our ILP formulation (*.lp* file) and the SAT (or the reduced SAT) formulation (*.cnf* file).

Our framework produces (i) the design rule-correct optimal routing solution in terms of wirelength and metal length through ILP-based detailed routing procedure; and (ii) a design rule-correct routing feasibility assessment through the SAT-based or the reduced SAT-based routability analysis procedure. Figures 4.9-4.11 respectively show an example pin layout on $M1$, the ILP-based routing solution, and the reduced SAT-based routing solution. In Figure 4.10, the optimal wirelength and metal length are different since there are five more metal segments (depicted in red-dotted circles) to avoid design rule violation (DRV). The reduced SAT-based solution in Figure 4.11 is not optimized, but takes 0.19% of ILP's runtime to analyze the design-rule correct routability.

4.5.2 Experimental Results

Our framework is validated on a 2.6GHz Intel Xeon E5-2640 Linux workstation with 128GB memory and 16 hyperthreaded CPU cores. For fair comparisons between the ILP-based routing optimization[23] and the SAT-based routability analysis[24], we restrict the maximum number of threads to 16 during experiments for both the ILP solver *CPLEX* [157] and the SAT solver *Plingeling* [163]. For *CPLEX*, we limit the maximum execution time to 12 hours to avoid tedious optimization procedure regardless of solution existence.[25]

[23] *CPLEX* provides user options to display the conflicts among variables/constraints and to check the feasibility of the problem. However, we observe that both procedures take longer than 12 hours for our smallest testcase, i.e., 14×13 with 60% pin density.

[24] A class of MaxSAT and MinSAT solvers suggest optimization features, however, their scalability is usually limited to thousands instances thus inapplicable to our framework.

[25] E.g., we run one of our 20_19_70 testcases in Table 4.2 after execution for a week using *CPLEX* 16 threads, however, the gap to the best node is still 12.28% (started with 15%).

Table 4.2: Experimental results presenting the ILP-based detailed routing vs. the SAT- and the Reduced SAT-based routability analysis. Each row represents results of five distinct testcases, and shows numbers on average.

Testcase	Spec.			ILP-based Routing Optimization							SAT-based Routability Analysis						Reduced SAT-based Routability Analysis						
	#N	#P	#O	#Var	#Con	RAI			RC		T(s)	#Var	#Lit	#Cla	RAS			T(s)	T%	#Var	#Lit	#Cla	RAS			T(s)	T%
14_13_60	10.0	21.8	7.8	144,721.2	232,174.2	5	0	0	276.6	0.0	517.3	144,721.2	47,257,881.8	23,622,776.2	5	0	0	39.5	8.63	80,315.6	6,807,435.0	3,471,788.6	5	0	0	3.1	**0.63**
14_13_70	12.0	26.4	9.4	195,522.0	312,683.8	4	1	0	364.0	0.0	1,408.0	195,522.0	80,870,759.4	40,435,281.0	4	1	0	96.8	9.99	107,112.6	12,046,890.0	6,048,136.6	4	1	0	3.7	**0.39**
14_13_80	12.8	29.0	10.0	222,476.0	359,912.4	0	5	0	-	-	660.4	222,476.0	96,445,155.4	48,224,657.2	0	5	0	93.8	16.92	121,437.6	14,507,663.0	7,246,112.6	0	5	0	4.5	**0.77**
14_13_90	14.6	32.6	10.6	262,910.4	419,867.8	0	5	0	-	-	721.0	262,910.4	140,237,382.8	70,125,695.6	0	5	0	98.4	14.65	142,522.8	22,517,142.2	11,179,371.8	0	5	0	5.1	**0.71**
14_19_60	14.2	32.2	10.2	357,111.8	578,829.2	4	1	0	486.8	5.0	23,263.3	357,111.8	242,613,850.6	121,311,501.8	4	1	0	403.4	4.86	194,278.6	27,316,733.8	13,743,476.8	4	1	0	7.3	**0.07**
14_19_70	16.4	36.0	11.0	417,015.6	663,016.0	4	1	0	596.8	7.6	31,855.7	417,015.6	339,870,052.0	169,947,617.8	4	1	0	619.9	4.23	225,426.4	38,228,830.0	19,143,867.2	4	1	0	9.8	**0.06**
14_19_80	18.6	41.8	12.8	536,195.4	858,173.4	0	3	2	-	-	18,869.2	536,195.4	516,238,692.2	258,153,515.2	0	5	0	813.8	20.25	287,674.2	62,528,212.2	31,102,552.6	0	5	0	12.0	**0.27**
14_19_90	19.8	44.6	12.6	566,157.6	903,384.4	0	3	2	-	-	19,097.6	566,157.6	583,624,359.2	291,844,830.6	0	5	0	709.9	13.00	303,519.2	70,335,765.8	34,962,629.8	0	5	0	12.9	**0.22**
20_13_60	14.6	32.8	10.8	368,255.8	595,007.8	4	1	0	489.5	0.0	11,465.1	368,255.8	258,382,327.2	129,202,420.4	4	1	0	436.8	5.54	199,829.8	29,761,672.6	14,947,286.4	4	1	0	7.1	**0.08**
20_13_70	16.4	36.2	11.2	415,530.2	662,946.0	3	1	1	571.0	5.3	37,227.0	415,530.2	355,116,364.0	167,578,851.2	3	2	0	439.8	1.33	224,503.8	38,550,581.4	19,287,022.2	3	2	0	9.2	**0.03**
20_13_80	17.6	40.0	11.0	453,804.8	732,491.4	0	4	1	-	-	17,771.1	453,804.8	375,531,847.4	187,778,810.2	0	5	0	639.7	5.35	245,035.2	42,641,520.6	21,266,387.2	0	5	0	8.1	**0.06**
20_13_90	19.4	44.2	12.2	539,266.0	869,006.2	0	4	1	-	-	18,634.8	539,266.0	513,571,393.8	256,814,507.6	0	5	0	763.9	8.27	289,517.0	61,999,434.0	30,794,434.8	0	5	0	10.9	**0.13**
20_19_60	16.6	44.8	12.8	797,507.6	1,293,681.0	0	1	4	-	-	41,849.2	797,507.6	1,072,225,606.0	536,164,726.6	3	1	1	3,043.3	8.00	427,871.0	95,090,666.4	47,596,732.8	4	1	0	24.8	**0.06**
20_19_70	23.0	52.6	14.6	1,037,195.2	1,677,742.8	0	1	4	-	-	39,398.3	1,037,195.2	1,749,427,423.0	874,816,269.0	0	2	3	468.9	1.48	552,123.0	176,174,452.4	87,585,849.8	0	5	0	31.5	**0.08**
20_19_80	25.0	56.8	13.8	1,065,155.6	1,711,462.2	0	5	0	-	-	26,490.7	1,065,155.6	1,897,303,801.4	948,737,208.4	0	1	4	462.1	1.20	567,544.8	174,902,017.8	87,039,386.6	0	5	0	29.1	**0.12**
20_19_90	28.6	64.4	15.4	1,298,447.2	2,073,363.0	0	3	2	-	-	30,353.6	1,298,447.2	2,694,707,009.0	1,347,486,502.6	0	0	5	-	-	688,238.2	255,009,623.2	126,674,314.2	0	5	0	42.5	**0.16**
SD%	5.5	4.7	15.9	15.1	15.0	-	-	-	4.2	9.8	45.9	15.1	25.4	25.5	-	-	-	27.1	-	14.5	31.8	31.4	-	-	-	28.9	-
T% Average				(reference) **100.00**															**8.25**								**0.24**

Table 4.2 presents our experimental results comparing the ILP-based detailed routing vs. the SAT- and the reduced SAT-based routability analysis. We validate our implementation with 80 testcases. Note that each row in Table 4.2 represents five distinct testcases, and shows numbers on average. All routability analyses produce the design rule-correct routability assessments. The naming convention for the testcase in each row is $< 1 >\_< 2 >\_< 3 >$, where $<$ 1-3 $>$ are the numbers of vertical tracks, horizontal tracks, and pin density, respectively. In the table, #N = #Nets, #P = #Pins, #O = #Outer pins, #Var = #Variables, #Con = #Constraints, RAI = Routability Analysis by the ILP (#Routable, #Unroutable, and #Undecided testcases by the pre-specified time limit, i.e., 12 hours), RC = for routed testcases (average Routing Cost, average gap (%) between the best-feasible solution found and the best node when *CPLEX* is terminated by the time limit), T = runTime, #Lit = #Literals, #Cla = #Clauses, RAS = Routability Analysis by the SAT (or the reduced SAT) (#Routable, #Unroutable, and #Undecided testcases by tool limitation), and T% = runTime% vs. ILP. For the two bottom rows, SD% = the average Standard Deviation (in percent) for each column, T% Average = the average T% of the SAT and the reduced SAT. The symbol '-' denotes the empty entries.

For the ILP, our testcases range up to 1.5M of ILP variables and 2.5M of ILP constraints. For the SAT and the reduced SAT, the numbers of variables, literals and clauses range up to 1.6M, 3.4B and 1.7B, and 0.8M, 0.4B and 0.2B, respectively. For RAI and RAS (routability analysis by ILP and SAT), we report the numbers of routable, unroutable, and undecided (in terms of routability) testcases, respectively. We have undecided testcases due to the time limit of ILP procedure and the tool limitation of SAT solver.[26] For the ILP, if the testcase is routable, we report the routing cost (i.e., the total sum of routed wirelength and the metal length), and we separately show the gap% between the current best-feasible solution (at the moment of the termination) and the best node of optimization if *CPLEX* has any feasible solutions. At the two bottom rows, we present the average standard deviation in percent for each column and the average runtime% of the SAT and the reduced SAT.

Our reduced SAT-based routability analysis identifies the design rule-correct routing feasibility within 0.24% of the ILP runtime on average (ranging 0.01-1.05%). Among

[26] *Plingeling* can solve SAT problems with up to 0.6B clauses.

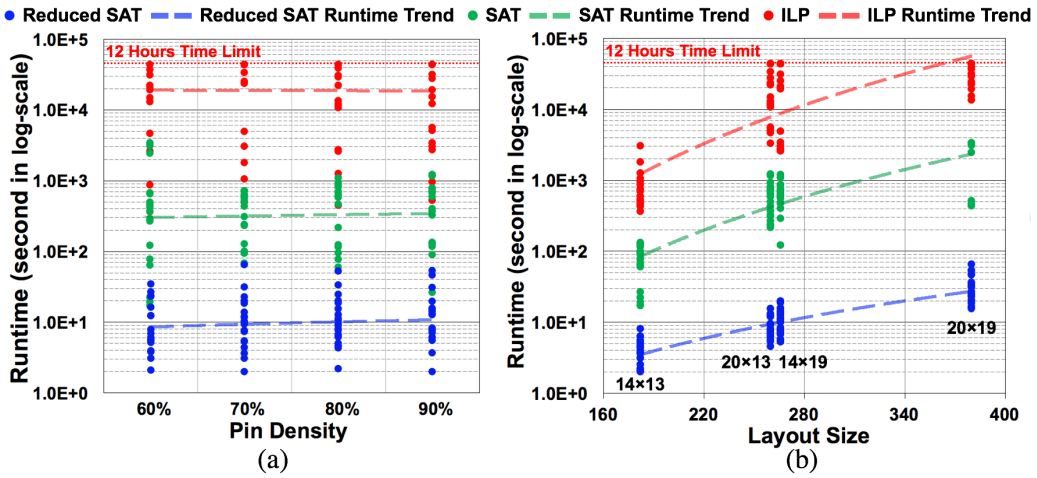


Figure 4.12: The runtime trends (in log-scale) across the (a) pin density and (b) layout size (#vertical $\times$ #horizontal tracks).

80 testcases, 28 testcases are routable. Across the pin density, 17 and 11 out of 20 testcases are routable at 60% and 70%, respectively. None of testcases are routable beyond 80%. As we do not apply the placement refinements for improving pin accessibility, we have a few unrouted testcases at the low pin densities (e.g., 60-70%) since they have locally congested regions. The ILP-, SAT-, and reduced SAT-based methods give the same routability assessment if they have solutions. While the reduced SAT-based analysis provides the routability results of all 80 testcases, the ILP-based and the SAT-based analyses fail to identify the routability for 17 and 13 out of 80 testcases, respectively. For ILP, 23 testcases are terminated by the time limit (six of them are routable). Figure 4.12 shows the runtime trends of our framework in log-scale across the (a) pin density and (b) layout size (i.e., #vertical $\times$ #horizontal tracks), indicating the reduced SAT-based routability analysis performs $> 1e^3\times$ faster than the ILP-based analysis.

4.6 Conclusion

We have described our framework, a new design methodology for fast turnaround in analyzing the routing feasibility of the given layout architecture. Our framework efficiently identifies the design rule-correct routability by creating the well-organized ILP and SAT formulation. We develop a new SAT-friendly ILP-based detailed routing formu-

lation satisfying conditional design rules. During our ILP-to-SAT conversion, we reduce the complexity of the SAT problem by utilizing a logic minimizer and further refining the SAT formula. We demonstrate that our resulting SAT-based routability analysis produces the precise assessment of the design rule-correct routability, within 0.24% of ILP runtime on average.

Our future work includes (i) to directly derive the SAT formulation, (ii) to further reduce the SAT's complexity, (iii) to diagnose potentail routing failures by extracting conflicting clauses, and (iv) to develop a new design tool to plan routable layout patterns for optimal IC chip area usage.

4.7 Acknowledgements

Chapter 4 contains a reprint of Ilgweon Kang, Dongwon Park, Changho Han and Chung-Kuan Cheng, "Fast and Precise Routability Analysis with Conditional Design Rules", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2018. The dissertation author is the primary author of the paper.

I would like to thank my coauthors Dongwan Park, Changho Han and Chung-Kuan Cheng. Also I would like to thank Professor Masahiro Fujita and Professor Sicun Gao for invaluable comments and discussions.

Chapter 5

Conclusion

This thesis describes new design methodologies for advanced technology nodes in three categories: (1) 3-D IC floorplan representations, (2) constraint- and routability-driven global placement, and (3) conditional design rule-correct routability analysis.

Chapter 2 presents new 3-D IC floorplan representations, i.e., *corner links*, *four trees*, and *partial order*, enhancing 3-D IC physical design automation. As the market demand smaller footprint/wirelength and less power consumption, but better performance and more functionalities, three-dimensional integrated circuit (3-D IC) offers a potential breakthrough to enable a paradigm-shift strategy. Algorithmic 3-D IC floorplan representations describe orientations and physical positions of each block relative to the origin in the 3-D space, also provide a foundation of data structure to efficiently search 3-D IC's design space. Chapter 2 introduces a novel 3-D IC floorplan representation, called *corner links*, analyzes the relation of the corner links to their corresponding *partial order* representation, and discusses the equivalence relation of the corner links and the partial order representations through mathematical proofs and algorithms. The chapter dissects several key properties. First, the corner links representation can be reduced to its corresponding partial order representation. Second, *four trees* representation equivalently expresses the corner links representation of the non-degenerate 3-D mosaic floorplan. Third, the 3-D mosaic floorplan is valid if the partial order representation defines relationships between all pairs of blocks. Last, the partial order representation captures all cutting planes in the 3-D mosaic floorplan, in the order of their respective dimensions. The chapter also show

that the partial order representation can restore the absolute coordinates of all blocks in the 3-D mosaic floorplan by using the given physical dimensions of blocks.

Chapter 3 describes our constraint- and routability-driven global placement engine, *RePlAce*. RePlAce is a flat, nonlinear analytical global placement engine with electrostatics-based global-smooth density cost function, addressing routing congestion as well as classical design goals with analogy of charge and electrical potential distribution. Chapter 3 proposes a new constraint-oriented local-density function for mixed-size placement that incorporates (i) a constraint-oriented local-density penalty factor for each bin, and (ii) a constraint-oriented local-density cost coefficient for each instance. The chapter also proposes a methodology for density-penalty adaptation via an improved *dynamic step size adaptation* that automatically adjusts the density penalty factor based on the *HPWL curve* (i.e., trajectory of HPWL cost versus iteration count) observed in a *trial placement procedure*. The improved dynamic step size adaptation applies more fine-grained control at transition points on the HPWL curve. RePlAce is validated by HPWL comparison to all best known ISPD and MMS benchmark results. Without any testcase-specific tuning, RePlAce achieves an average HPWL reduction of 2.00% over the best known ISPD benchmark results, and of 2.73% over the best known MMS benchmark results. For routability-driven global placement, the chapter proposes a layer-aware cell inflation technique, considering per-layer pin blockages. RePlAce integrates the official global router *NCTU-GR* [161] of the DAC-2012 and ICCAD-2012 benchmark suites for congestion estimation. RePlAce uses a simple but effective superlinear cell inflation technique to mitigate global routing congestion during global placement. Following the strategy of recent leading works [42] [43], a post-placement optimization by [76] is included. By integrating all innovations to improve routability, RePlAce delivers solution quality in terms of scaled HPWL that substantially improves over previous leading academic placers for the DAC-2012 and ICCAD-2012 benchmark suites, achieving on average 8.50% to 9.59% scaled HPWL reduction over previous placers. RePlAce is the first work to achieve superior solution quality across all the ISPD-2005, ISPD-2006, MMS, DAC-2012 and ICCAD-2012 benchmark suites with a single placement engine.

Chapter 4 presents a new design methodology that efficiently identifies the conditional design rule-correct routability through well-organized ILP (integer linear programming) and SAT (Boolean satisfiability) formulation. Since scaling metal pitch is lagged

behind scaling device pitch, securing pin accessibility of IC components has become a critical bottleneck during detailed routing, e.g., less number of routing tracks, increasing pin density, and smaller pin geometry make the pin accessibility problem harder. The absence of fast and precise routability analysis tool at the early physical design stage (e.g., before routing) exposes the entire IC design project to high-risk unpredictability. Chapter 4 proposes a fast and precise routability analysis framework by using a novel SAT formulation comprehending the net structure (i.e., source-sink connectivity) and the conditional design rules. The proposed framework produces *design rule-correct routability assessment*, offering an early "go/no-go" decision opportunity for the remaining PD procedure. In the chapter, a new *SAT-friendly ILP* formulation is proposed based on multi-commodity flow. In contrast to previous ILP-based detailed routers, the proposed ILP formulation refines all variables at the solution space as 0-1 indicator (so-called *SAT-friendly*) and defines source-sink connectivity of each net at a per-commodity granularity, to facilitate the SAT conversion. Also an *ILP-to-SAT conversion* mechanism is developed which exploits CNF (conjunctive normal form) formula by a logic minimizer [152] upon dissecting the ILP formulation. This reduces the number of literals and clauses for less complexity to further enhance runtime of the SAT solver [163]. Our resulting framework performs the routability analysis *within 0.24% of ILP runtime on average*, while guaranteeing the precise assessment of the design rule-correct routability.

Bibliography

[1] R. Aitken, G. Yeric, B. Cline, S. Sinha, L. Shifren, I. Iqbal and V. Chandra, "Physical Design and FinFETs", *Proc. ACM/IEEE International Symposium on Physical Design*, 2014, pp. 65-68.

[2] F. A. Aloul, A. Ramani, I. L. Markov and K. A. Sakallah, "Generic ILP versus Specialized 0-1 ILP: An Update", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2002, pp. 450-457.

[3] C. J. Alpert, "The ISPD98 Circuit Benchmark Suite", *Proc. ACM/IEEE International Symposium on Physical Design*, 1998, pp. 80-85.

[4] C. J. Alpert, Z. Li, M. D. Moffitt, G.-J. Nam, J. A. Roy and G. Tellez, "What Makes a Design Difficult to Route", *Proc. ACM/IEEE International Symposium on Physical Design*, 2010, pp. 7-12.

[5] C. J. Alpert, Z. Li, G.-J. Nam, C. N. Sze, N. Viswanathan and S. I. Ward, "Placement: Hot or Not?", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2012, pp. 283-290.

[6] K. J. Antreich, F. M. Johnnes and F. H. Kirsch, "A New Approach for Solving the Placement Problem using Force Models",' *Proc. IEEE International Symposium on Circuits and Systems*, 1982, pp. 481-486.

[7] Y. Badr, K.-W. Ma and P. Gupta, "Layout Pattern-Driven Design Rule Evaluation", *SPIE Journal of Micro/Nanolithography, MEMS, and MOEMS* 13(4) (2014), pp. 043018:1-043018:8.

[8] M. G. Bardon, Y. Sherazi, P. Schuddinck, D. Jang, D. Yakimets, P. Debacker, R. Baert, H. Mertens, M. Badaroglu, A. Mocuta, N. Horiguchi, D. Mocuta, P. Raghavan, J. Ryckaert, A. Spessot, D. Verkest and A. Steegen, "Extreme Scaling Enabled by 5 Tracks Cells: Holistic Design-Device Co-optimization for FinFETs and Lateral Nanowires", *Proc. IEEE International Electron Devices Meeting*, 2016, pp. 28.2.1-28.2.4.

[9] G. Baxter, "On Fixed Points of the Composite of Commuting Functions", *Journal of the American Mathematical Society* 15(6) (1964), pp. 851-855.

[10] K. Bernstein, P. Andry, J. Cann, P. Emma, D. Greenberg, W. Haensch, M. Ignatowski, S. Koester, J. Magerlein, R. Puri and A. Young, "Interconnects in the Third Dimension: Design Challenges for 3D ICs", *Proc. ACM/IEEE Design Automation Conference*, 2007, pp. 562-567.

[11] I. S. Bustany, D. Chinnery, J. R. Shinnerl and V. Yutsis, "ISPD 2015 Benchmarks with Fence Regions and Routing Blockages for Detailed-Routing-Driven Placement", *Proc. ACM/IEEE International Symposium on Physical Design*, 2015, pp. 157-164.

[12] U. Brenner and A. Rohe, "An Effective Congestion-Driven Placement Framework", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 22(4) (2003), pp. 387-394.

[13] F. Brglez, D. Bryan and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits", *Proc. IEEE International Symposium on Circuits and Systems*, 1989, pp. 1929-1934.

[14] H. H. Chan, S. N. Adya and I. L. Markov, "Are Floorplan Representations Important in Digital Design?", *Proc. ACM/IEEE International Symposium on Physical Design*, 2005, pp. 129-136.

[15] T. F. Chan, J. Cong, M. Romesis, J. R. Shinnerl, K. Sze and M. Xie, "mPL6: A Robust Multilevel Mixed-Size Placement Engine", *Proc. ACM/IEEE International Symposium on Physical Design*, 2005, pp. 227-229.

[16] W.-T. J. Chan, Y. Du, A. B. Kahng, S. Nath and K. Samadi, "3D-IC Benefit Estimation and Implementation Guidance from 2D-IC Implementation", *Proc. ACM/IEEE Design Automation Conference*, 2015, pp. 30:1-30:6.

[17] W.-T. J. Chan, P.-H. Ho, A. B. Kahng and P. Saxena, "Routability Optimization for Industrial Designs at Sub-14nm Process Nodes Using Machine Learning", *Proc. ACM/IEEE International Symposium on Physical Design*, 2017, pp. 15-21.

[18] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen and Y.-W. Chang, "A High-Quality Mixed-Size Analytical Placer Considering Preplaced Blocks and Density Constraints", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2006, pp. 187-192.

[19] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen and Y.-W. Chang, "NTUplace3: An Analytical Placer for Large-Scale Mixed-Size Designs with Preplaced Blocks and Density Constraints", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27(7) (2008), pp. 1228-1240.

[20] L. Cheng, L. Deng and M. D. F. Wong, "Floorplanning for 3-D VLSI Design", *Proc. ACM/IEEE Asia and South Pacific Design Automation Conference*, 2005, pp. 405-411.

[21] C.-K. Cheng, P. Du, A. B. Kahng and S.-H. Weng, "Low-Power Gated Bus Synthesis for 3D IC via Rectilinear Shortest-path Steiner Graph", *Proc. ACM/IEEE International Symposium on Physical Design*, 2012, pp. 105-112.

[22] C.-K. Cheng and E. S. Kuh, "Module Placement Based on Resistive Network Optimization", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 3(3), 1984, pp. 218-225.

[23] F. R. K Chung, R. L. Graham, V. E. Hoggatt Jr. and M. Kleiman, "The Number of Baxter Permutations", *Journal of Combinatorial Theory, Series A* 24(3) (1978), pp. 382-394.

[24] J. Cong, G. Luo, K. Tsota and B. Xiao, "Optimizing Routability in Large-scale Mixed-size Placement", *Proc. ACM/IEEE Asia and South Pacific Design Automation Conference*, 2013, pp. 441-446.

[25] J. Cong, G. Luo, J. Wei and Y. Zhang, "Thermal-Aware 3D IC Placement Via Transformation", *Proc. ACM/IEEE Design Automation Conference*, 2007, pp. 780-785.

[26] P. Cremer, S. Hougardy, J. Schneider and J. Silvanus, "Automatic Cell Layout in the 7nm Era", *Proc. ACM/IEEE International Symposium on Physical Design*, 2017, pp. 99-106.

[27] N. K. Darav, A. Kennings, A. F. Tabrizi, D. Westwick and L. Behjat, "Eh?Placer: A High-Performance Modern Technology-Driven Placer", *ACM Transactions on Design Automation of Electronic Systems* 21(3) (2016), pp. 37:1-37:27.

[28] P. Debacker, K. Han, A. B. Kahng, H. Lee, P. Raghavan and L. Wang, "Vertical M1 Routing-Aware Detailed Placement for Congestion and Wirelength Reduction in Sub-10nm Nodes", *Proc. ACM/IEEE Design Automation Conference*, 2017, pp. 1-6.

[29] Y. Ding, C. Chu and W.-K. Mak, "Pin Accessibility-Driven Detailed Placement Refinement", *Proc. ACM/IEEE International Symposium on Physical Design*, 2017, pp. 133-140.

[30] S. Dulucq and O. Guibert, "Baxter Permutations 1", *Discrete Mathematics* 180(1-3) (1998), pp. 143-156.

[31] R. Fischbach, J. Lienig and M. Thiele, "Solution Space Investigation and Comparison of Modern Data Structures for Heterogeneous 3D Designs", *Proc. IEEE International Conference of 3D System Integration*, 2010, pp. 1-8.

[32] H. Fraisse, A. Joshi, D. Gaitonde and A. Kaviani, "Boolean Satisfiability-Based Routing and Its Application to Xilinx UltraScale Clock Network", *Proc. ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2016, pp. 74-79.

[33] K. Fujiyoshi, H. Kawai and K. Ishihara, "A Tree Based Novel Representation for 3D-Block Packing", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28(5) (2009), pp. 759-764.

[34] B. Goplen and S. S. Sapatnekar, "Placement of 3D ICs with Thermal and Interlayer Via Considerations", *Proc. ACM/IEEE Design Automation Conference*, 2007, pp. 626-631.

[35] S. Goto, I. Cederbaum and B. S. Ting, "Suboptimum Solution of the Back-Board Ordering with Channel Capacity Constraint", *IEEE Transactions on Circuits and Systems* CAS-24(11), 1977, pp. 645-652.

[36] S. Goto and E. S. Kuh, "An Approach to the Two-Dimensional Placement Problem in Circuit Layout", *IEEE Transactions on Circuits and Systems* CAS-25(4), 1978, pp. 208-217.

[37] S. Goto, "A Two-Dimensional Placement Algorithm for the Master Slice LSI Layout Problem", *Proc. ACM/IEEE Design Automation Conference*, 1979, pp. 11-17.

[38] S. Goto, "An Efficient Algorithm for the Two-Dimensional Placement Problem in Electrical Circuit Layout", *IEEE Transactions on Circuits and Systems* CAS-28(1), 1981, pp. 12-18.

[39] S. Goto, T. Matsuda, K. Takamizawa, T. Fujita, H. Mizumura, H. Nakamura and F. Kitajima, "LAMBDA, an Integrated Master-Slice LSI CAD System", *Integration, the VLSI Journal* 1(1), 1983, Elsevier, pp. 53-69.

[40] K. Han, A. B. Kahng and H. Lee, "Evaluation of BEOL Design Rule Impacts Using An Optimal ILP-based Detailed Router", *Proc. ACM/IEEE Design Automation Conference*, 2015, pp. 1-6.

[41] X. He, T. Huang, L. Xiao, H. Tian, G. Cui and E. F. Y. Young, "Ripple: An Effective Routability-Driven Placer by Iterative Cell Movement", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2011, pp. 74-79.

[42] X. He, T. Huang, L. Xiao, H. Tian and E. F. Y. Young, "Ripple: A Robust and Effective Routability-Driven Placer", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32(10) (2013), pp. 1546-1556.

[43] X. He, Y. Wang, Y. Guo and E. F. Y. Young, "Ripple 2.0: Improved Movement of Cells in Routability-Driven Placement", *ACM Transactions on Design Automation of Electronic Systems* 22(1) (2016), pp. 10:1-10:26.

[44] X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.-K. Cheng and J. Gu, "Corner Block List: An Effective and Efficient Topological Representation of Non-Slicing Floorplan", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2000, pp. 8-12.

[45] M.-K. Hsu and Y.-W. Chang, "Unified Analytical Global Placement for Large-Scale Mixed-Size Circuit Designs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31(9) (2012), pp. 1366-1378.

[46] M. K. Hsu, Y.-W. Chang and V. Balabanov, "TSV-aware Analytical Placement for 3D IC Designs", *Proc. ACM/IEEE Design Automation Conference*, 2011, pp. 664-669.

[47] M.-K. Hsu, Y.-F. Chen, C.-C. Huang, S. Chou, T.-H. Lin, T.-C. Chen and Y.-W. Chang, "NTUplace4h: A Novel Routability-Driven Placement Algorithm for Hierarchical Mixed-Size Circuit Designs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33(12) (2014), pp. 1914-1927.

[48] M.-K. Hsu, N. Katta, H. Y.-H. Lin, K. T.-H. Lin, K. H. Tam and K. C.-H. Wang, "Design and Manufacturing Process Co-optimization in Nano-technology", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2014, pp. 574-581.

[49] X. Hu, P. Du, J. F. Buckwalter and C.-K. Cheng, "Modeling and Analysis of Power Distribution Networks in 3-D ICs", *IEEE Transactions on Very Large Scale Integration Systems* 21(2) (2013), pp. 354-366.

[50] J. Hu, A. B. Kahng, S. Kang, M. Kim and I. Markov, "Sensitivity-Guided Metaheuristics for Accurate Discrete Gate Sizing", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2012, pp. 233-239.

[51] T.-C. Hu and E.-S. Kuh, *VLSI Circuit Layout Theory and Design*, IEEE Press, 1985.

[52] J. Hu, J. A. Roy and I. L. Markov, "Completing High-Quality Global Routes", *Proc. ACM/IEEE International Symposium on Physical Design*, 2010, pp. 35-41.

[53] G. Huang, M. Bakir, A. Naeemi, H. Chen and J. D. Meindl, "Power Delivery for 3D Chip Stacks: Physical Modeling and Design Implication", *Proc. IEEE Electrical Performance of Electronic Packaging and Systems*, 2007, pp. 205-208.

[54] C.-C. Huang, C.-H. Chiou, K.-H. Tseng and Y.-W. Chang, "Detailed-Routing-Driven Analytical Standard-Cell Placement", *Proc. ACM/IEEE Asia and South Pacific Design Automation Conference*, 2015, pp. 378-383.

[55] X. Jia, Y. Cai, Q. Zhou, G. Chen, Z. Li and Z. Li, "MCFRoute: A Detailed Router Based on Multi-Commodity Flow Method" *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2014, pp. 397-404.

[56] Z.-W. Jiang, B.-Y. Su and Y.-W. Chang, "Routability-Driven Analytical Placement by Net Overlapping Removal for Large-Scale Mixed-Size Designs", *Proc. ACM/IEEE Design Automation Conference*, 2008, pp. 167-172.

[57] M. Jung, T. Song, Y. Wan, Y.-J. Lee, D. Mohapatra, H. Wang, G. Taylor, D. Jariwala, V. Pitchumani, P. Morrow, C. Webb, P. Fischer and S. K. Lim, "How to Reduce Power in 3D IC Designs: A Case Study with OpenSPARC T2 Core", *Proc. IEEE Custom Integrated Circuits Conference*, 2013, pp. 1-4.

[58] A. B. Kahng, "A Roadmap and Vision for Physical Design", *Proc. ACM/IEEE International Symposium on Physical Design*, 2002, pp. 112-117.

[59] A. B. Kahng, H. Lee and J. Li, "Horizontal Benchmark Extension for Improved Assessment of Physical CAD Research", *Proc. Great Lakes Symposium on Very Large Scale Integration*, 2014, pp. 27-32.

[60] A. B. Kahng, J. Lienig, I. L. Markov and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, Springer, 2011.

[61] A. B. Kahng, M. Luo, G.-J. Nam, S. Nath, D. Z. Pan and G. Robins, "Toward Metrics of Design Automation Research Impact", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2015, pp. 263-270.

[62] A. B. Kahng, S. Reda and Q. Wang, "APlace: A General Analytic Placement Framework", *Proc. ACM/IEEE International Symposium on Physical Design*, 2005, pp. 233-235.

[63] A. B. Kahng and Q. Wang, "Implementation and Extensibility of an Analytic Placer", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 24(5) (2005), pp. 734-747.

[64] I. Kang and C.-K. Cheng, "Physical Layout after Half a Century: From Back-Board Ordering to Multi-Dimensional Placement and Beyond", *Proc. ACM/IEEE International Symposium on Physical Design*, 2017, pp. 123-128.

[65] A. K. Khan, R. Vatsa, S. Roy and B. Das, "A New Efficient Topological Structure for Floorplanning in 3D VLSI Physical Design", *Proc. IEEE International Advance Computing Conference*, 2014, pp. 696-701.

[66] M.-C. Kim, J. Hu, D.-J. Lee and I. L. Markov, "A SimPLR method for Routability-Driven Placement", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2011, pp. 67-73.

[67] M.-C. Kim, J. Hu, J. Li and N. Viswanathan, "ICCAD-2015 CAD Contest in Incremental Timing-Driven Placement and Benchmark Suite", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2015, pp. 921-926. http://cad-contest.el.cycu.edu.tw/CAD-contest-at-ICCAD2015/index.html.

[68] M.-C. Kim, J. Hu and N. Viswanathan, "ICCAD-2014 CAD Contest in Incremental Timing-Driven Placement and Benchmark Suite", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2014, pp. 361-366. http://cad_contest.ee.ncu.edu.tw/CAD-Contest-at-ICCAD2014/default.html.

[69] M.-C. Kim, D.-J. Lee and I. L. Markov, "SimPL: An Effective Placement Algorithm", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31(1), 2012, pp. 50-60.

[70] M.-C. Kim and I. L. Markov, "ComPLx: An Competitive Primal-dual Lagrange Optimization for Global Placement", *Proc. ACM/IEEE Design Automation Conference*, 2012, pp. 747-752.

[71] M.-C. Kim, N. Viswanathan, Z. Li and C. Alpert, "ICCAD-2013 CAD contest in Placement Finishing and Benchmark Suite", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2013, pp. 268-270. http://cad-contest.cs.nctu.edu.tw/CAD-contest-at-ICCAD2013/default.html.

[72] J. M. Kleinhans, G. Sigl, F. M. Johannes and K. J. Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 10(3), 1991, pp. 356-365.

[73] J. Knechtel and J. Lienig, "Physical Design Automation for 3D Chip Stacks – Challenges and Solutions", *Proc. ACM/IEEE International Symposium on Physical Design*, 2016, pp. 3-10.

[74] J. U. Knickerbocker, P. S. Andry, B. Dang, R. R. Horton, C. S. Patel, R. J. Polastre, K. Sakuma, E. S. Sprogis, C. K. Tsang, B. C. Webb and S. L. Wright, "3D Silicon Integration", *Proc. IEEE Electronic Components and Technology Conference*, 2008, pp. 538-543.

[75] T. Lin and C. Chu, "POLAR 2.0: An Effective Routability-Driven Placer", *Proc. ACM/IEEE Design Automation Conference*, 2014, pp. 1-6.

[76] W.-H. Liu, C.-K. Koh and Y.-L. Li, "Optimization of Placement Solutions for Routability", *Proc. ACM/IEEE Design Automation Conference*, 2013, pp. 1-9.

[77] W.-H. Liu, W.-C. Kao, Y.-L. Li and K.-Y. Chao, "Multi-Threaded Collision-Aware Global Routing with Bounded-Length Maze Routing", *Proc. ACM/IEEE Design Automation Conference*, 2010, pp. 200-205.

[78] W.-H. Liu, W.-C. Kao, Y.-L. Li and K.-Y. Chao, "NCTU-GR 2.0: Multithreaded Collision-Aware Global Routing with Bounded-Length Maze Routing", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32(5) (2013), pp. 709-722.

[79] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng and C.-K. Cheng, "ePlace: Electrostatics Based Placement Using Nesterov's Method", *Proc. ACM/IEEE Design Automation Conference*, 2014, pp. 1-6.

[80] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng and C.-K. Cheng, "ePlace: Electrostatics-Based Placement Using Fast Fourier Transform and Nesterov's Method", *ACM Transactions on Design Automation of Electronic Systems* 20(2) (2015), pp. 17:1-17:34.

[81] J. Lu, H. Zhang, P. Chen H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng and C.-K. Cheng, "ePlace-MS: Electrostatics-Based Placement for Mixed-Size Circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34(5) (2015), pp. 685-698.

[82] J. Lu, H. Zhuang, I. Kang, P. Chen and C.-K. Cheng, "ePlace-3D: Electrostatics based Placement for 3D-ICs", *Proc. ACM/IEEE International Symposium on Physical Design*, 2016, pp. 11-18.

[83] K. Lucas, C. Cork, B. Yu, G. Luk-Pat, B. Painter and D. Z. Pan, "Implications of Triple Patterning for 14nm Node Design and Patterning", *SPIE Advanced Lithography*, 2012, pp. 1-12.

[84] L. Luo and M. D. F. Wong, "Ordered Escape Routing Based on Boolean Satisfiability", *Proc. ASPDAC*, 2008, pp. 244-249.

[85] Y. Ma, X. Hong, S. Dong and C.-K. Cheng, "3D CBL: An Efficient Algorithm for General 3D Packing Problems", *Proc. IEEE International Midwest Symposium on Circuits and Systems*, 2005, pp. 1079-1082.

[86] Y. Ma, J. Sweis, H. Yoshida, Y. Wang, J. Kye and H. J. Levinson, "Self-Aligned Double Patterning (SADP) Compliant Design Flow", *SPIE Advanced Lithography*, 2012, pp. 1-13.

[87] M. Marek-Sadowska and S. P. Lin, "Timing Driven Placement", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 1989, pp. 94-97.

[88] I. L. Markov, J. Hu and M.-C. Kim, "Progress and Challenges in VLSI Placement Research", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2012, pp. 275-282.

[89] I. L. Markov, J. Hu and M.-C. Kim, "Progress and Challenges in VLSI Placement Research", *Proc. of the IEEE* 103(11) (2015), pp. 1985-2003.

[90] S. Mukherjee and S. Roy, "SAT Based Multi Pin Net Detailed Routing for FPGA", *Proc. International Symposium on Electronic System Design*, 2010, pp. 141-146.

[91] H. Murata, K. Fujiyoshi, S. Nakatake and Y. Kajitani, "VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 15(12), 1996, pp. 1518-1524.

[92] G.-J. Nam, “ISPD 2006 Placement Contest: Benchmark Suite and Results”, *Proc. ACM/IEEE International Symposium on Physical Design*, 2006, pp. 167.

[93] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter and M. Yildiz, “The ISPD 2005 Placement Contest and Benchmark Suite”, *Proc. ACM/IEEE International Symposium on Physical Design*, 2005, pp. 216-220.

[94] G.-J. Nam, K. A. Sakallah and R. A. Rutenbar, “A New FPGA Detailed Routing Approach via Search-Based Boolean Satisfiability”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 21(6) (2002), pp. 674-684.

[95] R. H. J. M. Otten, “Automatic Floorplan Design”, *Proc. ACM/IEEE Design Automation Conference*, 1982, pp. 261-267.

[96] J. K. Ousterhout, “Corner Stitching: A Data-Structuring Technique for VLSI Layout Tools”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 3(1) (1984), pp. 87-100.

[97] J. K. Ousterhout, G. T. Hamachi, R. N. Mayo, W. S. Scott and G. S. Taylor, “The Magic VLSI Layout System”, *IEEE Design and Test of Computers* 2(1) (1985), pp. 19-30.

[98] M. M. Ozdal, “Detailed-Routing Algorithms for Dense Pin Clusters in Integrated Circuits”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28(3) (2009), pp. 340-349.

[99] M. M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke and C. Zhuo, “The ISPD-2012 Discrete Cell Sizing Contest and benchmark suite”, *Proc. ACM/IEEE International Symposium on Physical Design*, 2012, pp. 161-164.

[100] M. Pan and C. Chu, ”FastRoute: a Step to Integrate Global Routing into Placement”, *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2006, pp. 464-471.

[101] S. Panth, K. Samadi, Y. Du and S. K. Lim, “Design and CAD methodologies for low power gate-level monolithic 3D ICs”, *Proc. ACM/IEEE International Symposium on Low Power Electronic Design*, 2008, pp. 171-176.

[102] B. T. Preas and W. M. van Cleemput, “Placement Algorithms for Arbitrarily Shaped Blocks”, *Proc. ACM/IEEE Design Automation Conference*, 1979, pp. 474-480.

[103] F. Qiao, I. Kang, D. Kane, E. F. Y. Young, C.-K. Cheng and R. Graham, “3D Floorplan Representations: Corner Links and Partial Order”, *Proc. IEEE International Conference of 3D System Integration*, 2016, pp.1-5.

[104] N. Quinn and M. Breuer, "A Forced Directed Component Placement Procedure for Printed Circuit Boards", *IEEE Transactions on Circuits and Systems* 26(6), 1979, pp. 377-388.

[105] S. Russel and P. Norvig, *Artificial Intelligence - A Modern Approach*, Prentice Hall, 1995.

[106] N. Ryzhenko and S. Burns, "Standard Cell Routing via Boolean Satisfiability", *Proc. DAC*, 2012, pp. 603-612

[107] S. S. Sapatnekar, "Addressing Thermal and Power Delivery Bottlenecks in 3D Circuits", *Proc. ACM/IEEE Asia and South Pacific Design Automation Conference*, 2009, pp. 423-428.

[108] V. Satopää, J. Albrecht, D. Irwin and B. Raghavan, "Finding a "Kneedle" and a Haystack: Detecting Knee Points in System Behavior", *Proc. International Conference on Distributed Computing Systems*, 2011, pp. 166-171.

[109] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package", *IEEE Journal of Solid-State Circuits* 20(2), 1985, pp. 510-522.

[110] J. Seo, J. Jung, S. Kim and Y. Shin, "Pin Accessibility-Driven Cell Layout Redesign and Placement Optimization", *Proc. ACM/IEEE Design Automation Conference*, 2017, Article 54.

[111] A. Shayan, X. Hu, H. Peng, C.-K. Cheng, W. Yu, M. Popovich, T. Toms and X. Chen, "Reliability aware through silicon via planning for 3D stacked ICs", *Proc. ACM/IEEE Design, Automation and Test in Europe*, 2009, pp. 288-291.

[112] D. Shi and A. Davoodi, "Improving Detailed Routability and Pin Access with 3D Monolithic Standard Cells", *Proc. ACM/IEEE International Symposium on Physical Design*, 2017, pp. 107-112.

[113] G. Skollermo, "A Fourier Method for the Numerical Solution of Poisson's Equation", *AMS Journal on Mathematics of Computation* 29(131) (1975), pp. 697-711.

[114] P. Spindler and F. M. Johannes, "Fast and Accurate Routing Demand Estimation for Efficient Routability-Driven Placement", *Proc. ACM/IEEE Design, Automation and Test in Europe*, 2007, pp. 1226-1231.

[115] L. Steinberg, "The Backboard Wiring Problem: A Placement Algorithm," *SIAM Review* 3(1), 1961, pp. 37-50.

[116] J. E. Stevens, "Fast Heuristic Techniques for Placing and Wiring Printed Circuit Boards", *Ph. D. dissertation*, University of Illinois at Urbana-Champaign, 1972.

[117] T. Taghavi, Z. Li, C. Alpert, G.-J. Nam, A. Huber and S. Ramji, "New Placement Prediction and Mitigation Techniques for Local Routing Congestion", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2010, pp. 621-624.

[118] X. Tang and D. F. Wong, "FAST-SP: A Fast Algorithm for Block Placement Based on Sequence Pair", *Proc. ACM/IEEE Asia and South Pacific Design Automation Conference*, 2001, pp. 521-526.

[119] A. Thierer and A. Castillo, "Projecting the Growth and Economic Impact of the Internet of Things", *Technology Policy, Policy Briefing, Mercatus Center at George Mason University*, June 15, 2015, http://www.mercatus.org/system/files/IoT-EP-v3.pdf.

[120] N. Viswanathan, C. J. Alpert, C. N. Sze, Z. Li, G.-J. Nam and J. A. Roy, "The ISPD-2011 Routability-Driven Placement Contest and Benchmark Suite", *Proc. ACM/IEEE International Symposium on Physical Design*, 2011, pp. 141-146. http://cad_contest.ee.ncu.edu.tw/CAD-Contest-at-ICCAD2014/default.html.

[121] N. Viswanathan, C. J. Alpert, C. N. Sze, Z. Li and Y. Wei, "The DAC 2012 Routability-driven Placement Contest and Benchmark Suite", *Proc. ACM/IEEE Design Automation Conference*, 2012, pp. 774-782, http://archive.sigda.org/dac2012/contest/dac2012_contest.html.

[122] N. Viswanathan, C. J. Alpert, C. N. Sze, Z. Li and Y. Wei, "ICCAD-2012 CAD Contest in Design Hierarchy Aware Routability-Driven Placement and Benchmark Suite", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2012, pp. 345-348, http://cad-contest.cs.nctu.edu.tw/CAD-contest-at-ICCAD2012/problems/p2/p2.html.

[123] N. Viswanathan, M. Pan and C. Chu, "FastPlace 3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control", *Proc. ACM/IEEE Asia and South Pacific Design Automation Conference*, 2007, pp. 135-140.

[124] M. M. Waldrop, "The Chips are Down for Moore's Law", *Nature* 530(7589) (2016), pp. 144-147.

[125] C.-K. Wang, C.-C. Huang, S. S.-Y. Liu, C.-Y. Chin, S.-T. Hu, W.-C. Wu and H.-M. Chen, "Closing the Gap between Global and Detailed Placement: Techniques for Improving Routability", *Proc. ACM/IEEE International Symposium on Physical Design*, 2015, pp. 149-156.

[126] M. Wang, X. Yang, K. Eguro and M. Sarrafzadeh, "Multicenter Congestion Estimation and Minimization During Placement", *Proc. ACM/IEEE International Symposium on Physical Design*, 2000, pp. 147-152.

[127] R. Wang, E. F. Y. Young and C.-K. Cheng, "Representing Topological Structures for 3-D Floorplanning", *Proc. IEEE International Conference of Communications, Circuits and Systems*, 2009, pp. 1098-1102.

[128] R. Wang, E. F. Y. Young and C.-K. Cheng, "Complexity of 3-D Floorplans by Analysis of Graph Cuboidal Dual Hardness", *ACM Transactions on Design Automation of Electronic Systems* 15(4) (2010), pp. 33:1-33:22.

[129] R. Wang, E. F. Y. Young, Y. Zhu, F. C. Graham, R. Graham and C.-K. Cheng, "3-D Floorplanning Using Labeled Tree and Dual Sequences", *Proc. ACM/IEEE International Symposium on Physical Design*, 2008, pp. 54-59.

[130] M. B. Weindling, "A Method for the Best Geometric Placement of Units on a Plane", *Proc. ACM/IEEE Design Automation Conference*, 1964, pp. 5.1-5.54.

[131] J. Westra, C. Bartels and P. Groeneveld, "Probabilistic Congestion Prediction", *Proc. ACM/IEEE International Symposium on Physical Design*, 2004, pp. 204-209.

[132] R. Widialaksono, R. B. R. Chowdhury, Z. Zhang, J. Schabel, S. Lipa, E. Rotenberg, R. Davis and P. Franzon, "Physical Design of a 3D-Stacked Heterogeneous Multi-Core Processor", *Proc. IEEE International Conference of 3D System Integration*, 2016, pp. 1-5.

[133] G. J. Wipfler, M. Wiesel and D. A. Mlynski, "A combined force and cut algorithm for hierarchical VLSI layout" *Proc. ACM/IEEE Design Automation Conference*, 1982, pp. 671-677.

[134] D. F. M. Wong and C.-L. Liu "A New Algorithm for Floorplan Design", *Proc. ACM/IEEE Design Automation Conference*, 1986, pp. 101-107.

[135] X. Xu, B. Cline, G. Yeric, B. Yu and D. Z. Pan, "Self-aligned Double Patterning Aware Pin Access and Standard Cell Layout Co-optimization", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34(5) (2015), pp. 699-712.

[136] X. Xu, B. Yu, J.-R. Gao, C.-L. Hsu and D. Z. Pan, "PARR: Pin-Access Planning and Regular Routing for Self-Aligned Double Patterning", *ACM Transactions on Design Automation of Electronic Systems* 21(3) (2016), pp. 42:1-42:21.

[137] H. Yamazaki, K. Sakanushi, S. Nakatake and Y. Kajitani, "The 3D-Packing by Meta Data Structure and Packing Heuristics", *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 83(4) (2000), pp. 639-645.

[138] J. Z. Yan, N. Viswanathan and C. Chu, "Handling Complexities in Modern Large-Scale Mixed-Size Placement", *Proc. ACM/IEEE Design Automation Conference*, 2009, pp. 436-441.

[139] S. Yang, A. Gayasen, C. Mulpuri, S. Reddy and R. Aggarwal, "Routability-Driven FPGA Placement Contest", *Proc. ACM/IEEE International Symposium on Physical Design*, 2016, pp. 139-143.

[140] X. Yang, R. Kastner and M. Sarrafzadeh, "Congestion Estimation During Top-Down Placement", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 21(1) (2002), pp. 72-80.

[141] B. Yao, H. Chen, C.-K. Cheng and R. Graham, "Floorplan Representations: Complexity and Connections", *ACM Transactions on Design Automation of Electronic Systems* 8(1) (2003), pp. 55-80.

[142] E. F. Y. Young, C. C. N. Chu and Z. C. Shen, "Twin Binary Sequences: A Nonredundant Representation for General Nonslicing Floorplan", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 22(4) (2003), pp. 457-469.

[143] P.-H. Yuh, C. C.-Y. Lin, T.-W. Huang, T.-Y. Ho, C.-L. Yang and Y.-W. Chang, "A SAT-based Routing Algorithm for Cross-Referencing Biochips", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2011, pp. 6:1-6:7.

[144] P.-H. Yuh, C.-L. Yang and Y.-W. Chang, "Temporal Floorplanning Using the T-Tree Formulation", *Proc. ACM/IEEE International Conference on Computer-Aided Design*, 2004, pp. 300-305.

[145] P.-H. Yuh, C.-L. Yang and Y.-W. Chang, "Temporal Floorplanning Using the Three-Dimensional Transitive Closure subGraph", *ACM Transactions on Design Automation of Electronic Systems* 12(4) (2007), pp. 37:1-37:34.

[146] V. Yutsis, I. S. Bustany, D. Chinnery, J. Shinnerl and W.-H. Liu, "ISPD 2014 Benchmarks with Sub-45nm Technology Rules for Detailed-Routing-Driven Placement", *Proc. ACM/IEEE International Symposium on Physical Design*, 2014, pp. 161-168.

[147] P. Zarkesh-Ha, J. A. Davis, W. Loh and J. D. Meindl, "Prediction of Interconnect Fan-Out Distribution using Rent's Rule", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2000, pp. 107-112.

[148] L. Zhang, S. Dong, X. Hong and Y. Ma, "A Fast 3D-BSG Algorithm for 3D Packing Problem", *Proc. IEEE International Symposium on Circuits and Systems*, 2007, pp. 2044-2047.

[149] W. Zhang, W. Yu, X. Hu, A. Shayan, A. E. Engin and C.-K. Cheng, "Predicting the Worst-Case Voltage Violation in a 3D Power Network", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2009, pp. 93-98.

[150] W. Zhu, J. Chen, Z. Peng and G. Fan, "Nonsmooth Optimization Method for VLSI Global Placement", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34(4) (2015), pp. 642-655.

[151] Cadence Design Systems, Inc., Sr Software Engineering Group Director, *personal communication*, 2017.

[152] Espresso, Logic Minimizer, http://embedded.eecs.berkeley.edu/pubs/downloads/espresso/.

[153] General Purpose FFT Package, http://www.kurims.kyoto-u.ac.jp/~ooura/fft.html.

[154] Global Semiconductor Industry Revenue from ICs 2009-2018, http://www.statista.com/statistics/519456/.

[155] Hardware Design Cost: Faster, Cooler, Simpler, could FD-SOI be Cheaper too?, https://www.semiwiki.com/forum/content/2991-faster-cooler-simpler-could-fd-soi-cheaper-too.html.

[156] Hybrid Memory Cube Consortium, http://www.hybridmemorycube.org/.

[157] IBM ILOG CPLEX, http://www.ilog.com/products/cplex/.

[158] ISPD-2017 Contest, http://www.ispd.cc/contests/17/.

[159] ITRS Report 2015 Edition, http://www.semiconductors.org/main/2015_international_technology_roadmap_for_semiconductors_itrs/.

[160] Many Ways to Shrink: The Right Moves to 10 Nanometer and Beyond, https://staticwww.asml.com/doclib/investor/asml_3_Investor_Day-Many_ways_to_shrink_MvdBrink1.pdf.

[161] NCTU-GR, http://people.cs.nctu.edu.tw/~whliu/NCTU-GR.htm.

[162] OpenCores, http://opencores.org/projects/.

[163] Plingeling, Multi-Threading SAT Solver, http://fmv.jku.at/lingeling/.

[164] Semiconductor Sales Revenue Worldwide from 1987 to 2018, http://www.statista.com/statistics/266973/.

[165] Yole Développement, *Equipment & Materials for 3DIC & WLP Applications*, 2015, http://www.yole.fr/2014-galery-3D.aspx#I000350e1.