UNIVERSITY OF CALIFORNIA SAN DIEGO

Logical Reasoning Techniques for Physical Layout in Deep Nanometer Technologies

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

in

Electrical Engineering (Computer Engineering)

by

Dong Won Park

Committee in charge:

Professor Chung-Kuan Cheng, Chair
Professor Bill Lin, Co-Chair
Professor Sujit Dey
Professor Daniel Mertz Kane
Professor Patrick Philip Mercier

2021

The dissertation of Dong Won Park is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____
Co-Chair

_____
Chair

University of California San Diego

2021

DEDICATION

To my wife, Seolyn,

my daughter, Siyul Stella,

my son, Siwoo,

and my family

EPIGRAPH

Man proposes, God disposes.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to first thank my advisor, Prof. Chung-Kuan Cheng for his encouragement, support and guidance during my Ph.D. I am extremely grateful for his understanding in all aspects of my life, both research and life. I would like to also thank my committee members, Prof. Bill Lin, Prof. Sujit Dey, Prof. Daniel Kane, and Prof. Patrick Mercier for their feedback and discussions related to this Ph.D. work. I would like to thank Professor Namkyoo Park for his academical support and encouragement toward research. A special thanks to Dr. Bong Hyun You for his guidance and encouraging me to pursue a Ph.D during my entire career in Samsung Display. I learned many skills that helped me perform well in my research and work under his guidance.

I would like to thank all my lab colleges in VLSI lab for their active collaboration, help, and all the good memories. I would also like to give special thanks to Ilgweon Kang and Daeyeal Lee. I want to sincerely thank all support from Samsung Display. My research was made possible by funding from Samsung Scholarship.

Most importantly, I owe so much to my wife, Seolyn Ban, my daughter, Siyul Stella, my son, Siwoo, my parents and all my family. I would like to thank them for their endless love and support every day. I could overcome all the difficulties thanks to their understanding, patience, and love.

Chapters 2 and Chapters 3 contain materials from following publications: “Grid-based Framework for Routability Analysis and Diagnosis with Conditional Design Rules”, by Dongwon Park, Daeyeal Lee, Ilgweon Kang, Chester Holtz, Sicun Gao, Bill Lin and Chung-Kuan Cheng, which appears in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, February 2020. “ROAD: Routability analysis and diagnosis framework based on SAT technique”, by Dongwon Park, Ilgweon Kang, Yeseong Kim, Sicun Gao, Bill Lin, and Chung-Kuan Cheng, which appears in International Symposium on Physical Design, April 2019. The dissertation author was the primary investigator and author of these papers.

Chapters 4 contains materials from “SP&R: Simultaneous Placement and Routing framework for standard cell synthesis in sub-7nm”, by Dongwon Park, Daeyeal Lee, Ilgweon Kang, Sicun Gao, Bill Lin, and Chung-Kuan Cheng, which appears in IEEE/ACM Asia and South Pacific Design Automation Conference, January 2020. The dissertation author was the primary investigator and author of this paper.

Chapters 5 contains materials from following publications: “Standard-Cell Scaling Framework with Guaranteed Pin-Accessibility”, by Chung-Kuan Cheng, Daeyeal Lee, and Dongwon Park, which appears in IEEE International Symposium on Circuits and Systems, October 2020. “SP&R: SMT-based Simultaneous Place-&-Route for Standard Cell Synthesis of Advanced Nodes”, by Daeyeal Lee, Dongwon Park, Chiatung Ho, Ilgweon Kang, Hayoung Kim, Sicun Gao, Bill Lin, and Chung-Kuan Cheng, which appears in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, November 2020. The dissertation author was the primary investigator and author of this paper.

VITA

2003	B. S. in Electrical Engineering, Seoul National University, Korea
2003-2006	Application Software Engineer, Techserver Co., Korea
2009	M. S. in Electrical Engineering, Seoul National University, Korea
2009-2012	Engineer, Samsung Electronics Co., Ltd, Korea
2012-Present	Staff Engineer, Samsung Display Co., Ltd, Korea
2021	Ph. D. in Electrical Engineering, University of California San Diego, US

PUBLICATIONS

Chung-Kuan Cheng, Andrew B. Kahng, Hayoung Kim, Minsoo Kim, Daeyeal Lee, Dongwon Park, and Minkyu Woo, "PROBE2.0: A Systematic Framework for Routability Assessment from Technology to Design in Advanced Nodes", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, December 2020 (Under Revision) (Alphabetical Order)

Daeyeal Lee*, Dongwon Park*, Chiatung Ho, Ilgweon Kang, Hayoung Kim, Sicun Gao, Bill Lin, and Chung-Kuan Cheng, "SP&R: SMT-based Simultaneous Place-&-Route for Standard Cell Synthesis of Advanced Nodes", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, November 2020 *Contributed Equally

Chung-Kuan Cheng, Chiatung Ho, Daeyeal Lee, and Dongwon Park, "A Routability-Driven Complimentary-FET (CFET) Standard Cell Synthesis Framework using SMT", *2020 International Conference on Computer-Aided Design (ICCAD)*, November 2020 (Alphabetical Order)

Chung-Kuan Cheng, Daeyeal Lee, and Dongwon Park, "Standard-Cell Scaling Framework with Guaranteed Pin-Accessibility", *IEEE International Symposium on circuits and systems (ISCAS)*, October 2020 (Alphabetical Order)

Dongwon Park, Daeyeal Lee, Ilgweon Kang, Chester Holtz, Sicun Gao, Bill Lin, and Chung-Kuan Cheng, "Grid-based Framework for Routability Analysis and Diagnosis with Conditional Design Rules", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, February 2020

Dongwon Park*, Daeyeal Lee*, Ilgweon Kang, Sicun Gao, Bill Lin, and Chung-Kuan Cheng, "SP&R: Simultaneous Placement and Routing framework for standard cell synthesis in sub-7nm", *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, January 2020 *Contributed Equally

Dongwon Park, Ilgweon Kang, Yeseong Kim, Sicun Gao, Bill Lin, and Chung-Kuan Cheng, "ROAD : Routability Analysis and Diagnosis Framework Based on SAT Techniques", *International Symposium on Physical Design (ISPD)*, April 2019

Ilgweon Kang, Fang Qiao, Dongwon Park, Daniel Kane, Evangeline Fung Yu Young, Chung-Kuan Cheng, and Ronald Graham, "Three-dimensional Floorplan Representations by Using Corner Links and Partial Order", *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, December 2018

Pengwen Chen, Chung-Kuan Cheng, Dongwon Park, and Xinyuan Wang, "Transient Circuit Simulation for Differential Algebraic Systems using Matrix Exponential", *2018 International Conference on Computer-Aided Design (ICCAD)*, November 2018 (Alphabetical Order)

Ilgweon Kang, Dongwon Park, Changho Han, and Chung-Kuan Cheng, "Fast and precise routability analysis with conditional design rules", *System Level Interconnect Prediction Workshop (SLIP)*, June 2018

Chung-Kuan Cheng, Ronald Graham, Ilgweon Kang, Dongwon Park, and Xinyuan Wang, "Tree structures and algorithms for physical design", *International Symposium on Physical Design (ISPD)*, Mar 2018 (Alphabetical Order)

Xiang Zhang, Dongwon Park, and Chung-Kuan Cheng, "Boosting off-chip interconnects through chip-to-chip capacitive coupled communication", *IEEE Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, October 2017

BongHyun You, JaeHoon Lee, SeokHa Hong, Dongwon Park, HyunChang Kim, Deog-Kyoon Jeong, "Current Sensing Self-Compensation System for Gate Driving Circuit Employing Oxide TFTs", *Symposium Digest of Technical Papers (SID)*, May 2016

Bong Hyun You, Jae Sung Bae, Dongwon Park, Seok Ha Hong, Satoru Saito, Ju Tae Moon, "UD Resolution 240Hz LCD TV Display System with High Speed Driving", *Symposium Digest of Technical Papers (SID)*, June 2012

Bong Hyun You, Jae Sung Bae, Jai Hyun Koh, Dongwon Park, Heen Dol Kim, Kook Hwan Ahn, JiSun Kim, Seong Young Lee, SukWon Jung, Yun Jae Kim, Sung Tae Shin, SeokJin Han, Anthony Botzas, Sarah Hwang, Larissa Pan, MoonHwan Im, Candice Brown Elliott, "The Most PowerEfficient 11.6" Full HD LCD Using PenTile Technology for Notebook Application", *Symposium Digest of Technical Papers (SID)*, May 2010

Young Jin Jung, Dongwon Park, Sukmo Koo, Sunkyu Yu, Namkyoo Park, "Metal slit array Fresnel lens for wavelength-scale optical coupling to nanophotonic waveguides", *Optics express*, October 2009

Young Jin Jung, Dongwon Park, Sunkyu Yu, Sukmo Koo, Hyungsuk Yu, Sanghun Han, Namkyoo Park, Jae Hun Kim, Young Min Jhon, Seok Lee, "Metal-slit array fresnel-lens for optical coupling", *2009 International Conference on Numerical Simulation of Optoelectronic Devices(NUSOD)*, September 2009

ABSTRACT OF THE DISSERTATION

Logical Reasoning Techniques for Physical Layout in Deep Nanometer Technologies

by

Dong Won Park

Doctor of Philosophy in Electrical Engineering (Computer Engineering)

University of California San Diego, 2021

Professor Chung-Kuan Cheng, Chair
Professor Bill Lin, Co-Chair

As VLSI technologies are continuously evolving sub-10*nm*, design of the routable and manufacturable layout for integrated circuits (ICs) has been more challenging. To maintain power-performance-area-cost (PPAC) gains from many scaling barriers, IC design demands orchestrated innovations across the entire stages of the design-to-silicon infrastructure. For this design-technology co-optimization (DTCO) in each physical design stage, the holistic exploration is essential across all the design considerations due to the limited resources, high density, and complex conditional design rules. However, many conventional ways focus on divide-and-conquer-style sub-problems and/or heuristic approaches because of the huge search

space of the problem, resulting in limited optimality. In this dissertation, we propose several constraint-based exact solving, i.e., constraint satisfaction problem (CSP), frameworks in various physical design questions related to standard cell placement and routing such as detailed routing, standard cell synthesis, and engineering change order (ECO). Our outcomes have the enhanced optimality compared to conventional approaches due to the concurrent manner between design considerations without any sequential/separate procedures. We utilize/select the appropriate logical reasoning technique, such as Integer Linear Programming (ILP), Boolean satisfiability (SAT), and Satisfiability Modulo Theories (SMT), depending on the problem characteristics.

In detailed routing, routability (including pin-accessibility) between standard cells becomes a critical bottleneck due to the limited number of routing tracks, higher pin density, and complex design rules. To reduce turnaround time, we suggest a fast routability analysis framework to analyze routing feasibility by using SAT solving technology. Routability analysis framework produces design rule-correct routability assessment within only 0.02% of ILP runtime on average. Also, we propose a precise routability diagnosis framework to diagnose explicit reasons for design-rule violations (DRVs) in the form of human-interpretable explanations, while specifying conflicting design rules with a physical location.

To maximize PPAC gains in DTCO of standard cell synthesis, the automation of standard cell layout is essential for smooth technology transition. Since many conventional approaches lack the optimality of the cell layout due to the sequential/heuristic manner, we propose a SMT-based automated standard cell synthesis framework, which simultaneously solves place-and-route, through a novel dynamic pin allocation scheme without deploying any sequential/separate operations. After tackling the scalability by developing various search-space reduction techniques, our framework successfully generates a whole set of 7*nm* standard cell library. On top of complete cell libraries, we propose standard cell scaling framework which enables the parametric study of standard cell layout with respect to the scaled cell architectures. In particular, we strictly ensure the pin-accessibility of the cell layout, which is intrinsically restricted by limited track number,

through our novel Boolean constraints, while maintaining the scaling advantages.

Chapter 1

Introduction and Preliminaries

In the past decades, semiconductor technologies have significantly contributed to the human society, and led entire industries toward more advanced systems with evolution of integrated circuits (ICs). Innovations and advancements on physical design (PD) guide progresses of modern VLSI designs and the automation. By virtue of the PD community and their efforts, the power-performance-area-cost (PPAC) of ICs has been extremely enhanced, the overall industry has been profitable, and the global semiconductor market size has been grown dramatically for the past decades. Although innovations of VLSI technologies, semiconductor industry has failed to manage development (i.e., design and verification) cost due to the time-consuming turnaround time of iterative design procedures as shown in Figure 1.1. Design for the routable and manufacturable layout in advanced technologies is no more trivial due to complex design considerations such as limited resources, high density, and complex design rules.

As shown in Figure 1.2, allowable tracks of standard cell has been reduced from 9T at 20*nm* node in 2012 to 6T at 7*nm* node in 2019 and is predicted to be 5T or even below for 3*nm* node in the near future. To maintain PPAC gains from the scaling barriers, IC design demands orchestrated innovations across the entire manufacture stages of the design-to-silicon infrastructure. For this design-technology co-optimization (DTCO), the holistic exploration is

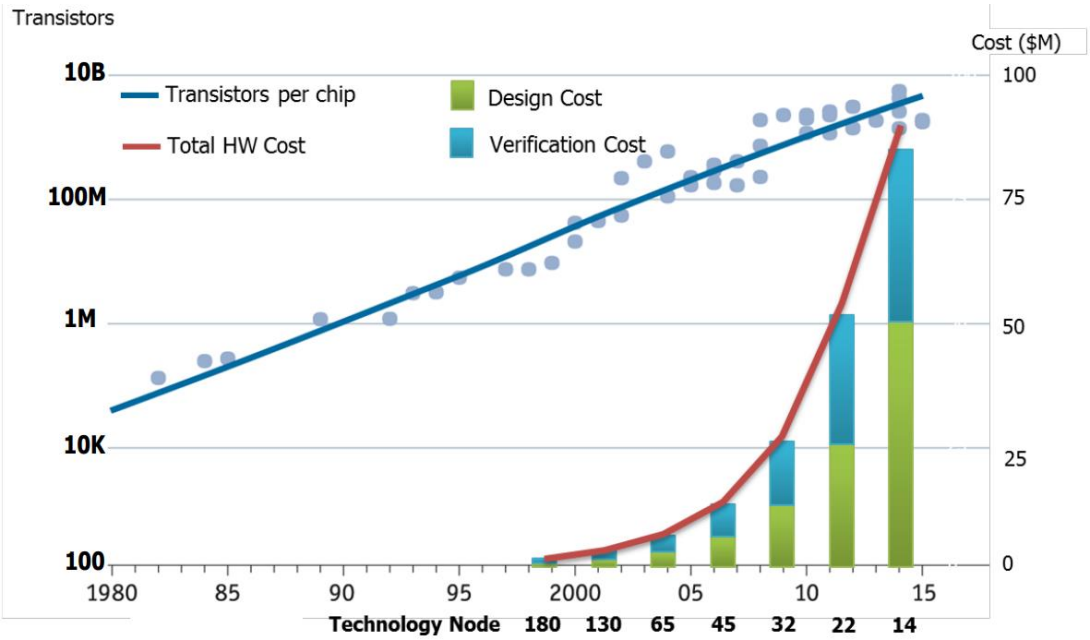


Figure 1.1: Design technology crisis [1]

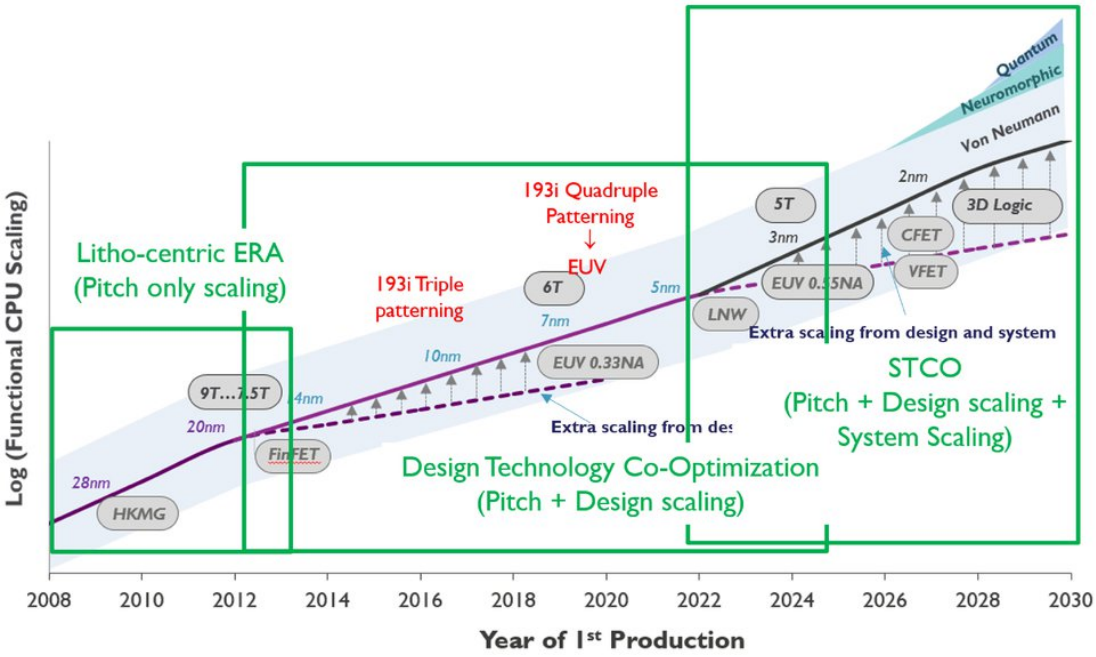


Figure 1.2: Scaling road map [2].

essential across all the design considerations of combined physical design stages. However, as we know, physical design for manufactureable IC layout is divided into many stages as depicted in Figure 1.3. This sequential design process is inevitable, because each subproblem

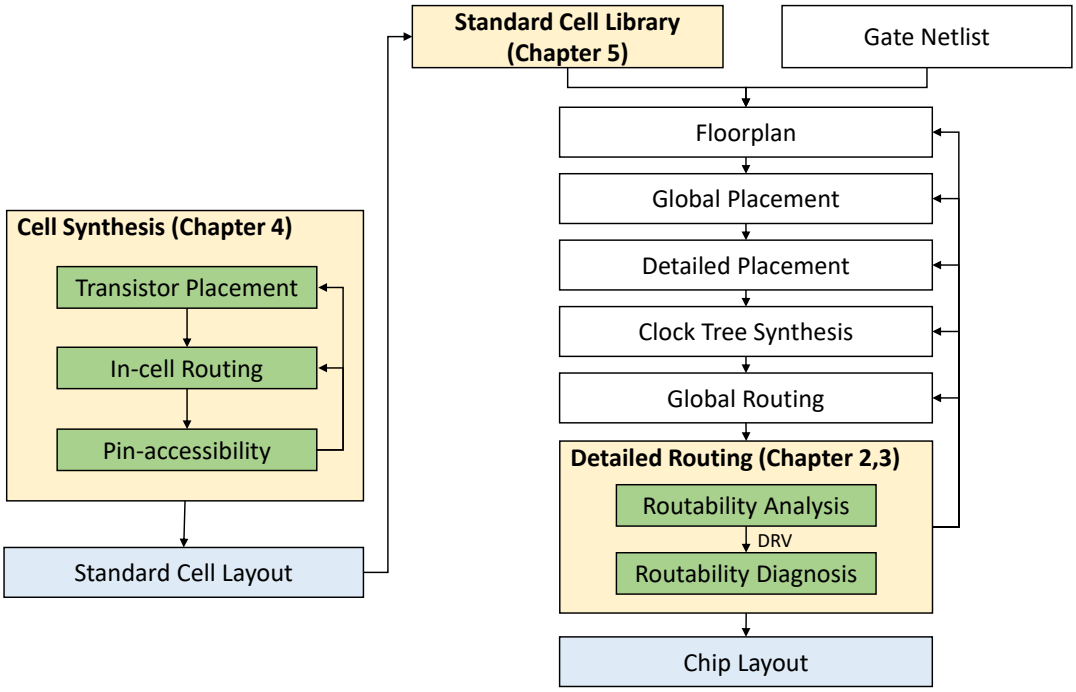


Figure 1.3: Physical design stages and the targets of this dissertation.

already requires an enormous search space due to the combinatorial searching for the millions of transistors. What was worse, some subproblem (e.g., placement or routing) is divided into further hierarchical/sequential sub-procedures such as global placement (GP), detailed placement (DP), global routing (GR), and detailed routing (DR). Therefore, many conventional works focus on divide-and-conquer-style and/or heuristic approaches due to the intrinsic scalability limitation of the problem. As a result, outcomes of these approaches are hard to reach the optimal solution due to the intractable search space partitioning and heuristic manner.

In this dissertation, we propose several exact solving methodologies and its automation for place-and-route problems in both in-cell (i.e., standard cell synthesis) and inter-cell (i.e., DR stage) across advanced technology nodes (i.e., sub-10nm), which have better optimality compared to the conventional approaches, through multi-objective optimization based on constraint-based design considerations. In other words, our problems (in both standard cell synthesis and detailed routing stage) are particular forms of constraint satisfaction problem (CSP), the subject of intense

research in artificial intelligence (AI). We convert practical DFM(Design for manufacturability)-related considerations in both placement and routing stages into either constraint or objectives to find better PPAC solutions. Then, for the resolution of CSP, we utilize the state-of-the-art solver of Integer Linear Programming (ILP), Boolean satisfiability (SAT), and Satisfiability Modulo Theories (SMT). To ensure the practical scalability of the framework from intrinsic complexity explosion of exact solving, we enrich/refine our constraints by developing efficient search space pruning techniques (e.g., breaking symmetry, conditional assignment using logical constraint), by adopting efficient representations (e.g., Boolean cardinality constraint and its encodings). As a consequence, we achieve the practical scalability of our framework for each target criteria. With this in mind, in the remaining Chapter 1, we introduce ILP, SAT, and SMT for its concept and several features/techniques utilized in this dissertation. Also, we describe basic foundations of our frameworks for placement and routing optimization. Then, this dissertation present new methodologies and its automated framework in three topics of physical design; (i) routability analysis and diagnosis in detailed routing (Chapter 2,3), (ii) standard cell synthesis (Chapter 4), and (iii) parametric standard cell scaling study for DTCO (Chapter 5)

The rest of this chapter is organized as follows. Section 1.1 introduces ILP and its usages in detailed routing. Section 1.2 introduces SAT and its techniques to refine the complexity of CNF (conjunctive normal form) representation. Section 1.3 introduces SMT and its general technique to reduce the solving complexity. Section 1.4 describes basic foundations of our frameworks. Table 1.1 presents the notations.

1.1 Integer Linear Programming (ILP) for Routing

Linear programming (LP) is a sort of optimization technique for a linear objective function, which are subject to linear equality and linear inequality constraints. If all variables in

[1]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 means FrontLeft.

Table 1.1: Notations for formulations in this dissertation.

Term	Description
X	The number of vertical tracks in the given routing (cell bounding) box
Y	The number of horizontal tracks in the given routing box
T	Set of FETs
t	t^{th} FET
ff_t	0-1 indicator if FET t is flipped
x_t	x-axis coordinate of lower-left corner of t
w_t (or h_t)	Width (or height) of FET t
P^t	Set of internal pins of FET t
p_i^t	i^{th} pin of FET t
$n(p)$	Net information of pin p
$G(V,E)$	Three-dimensional (3-D) routing graph
$V(V_i)$	Set of vertices in (i^{th} metal layer of) the routing graph G
v	A vertex with the coordinate (x_v, y_v, z_v)
v_d	A d-directional[1]adjacent vertex of v
$a(v)$	Set of adjacent vertices of v
$c_{v,u}$	Cost for metal segment on $e_{v,u}$
E	Set of edges in the routing graph G
$e_{v,u}$	An edge between v and u, $u \in a(v)$
$w_{v,u}$	Weighted cost for metal segment on $e_{v,u}$
N	Set of multi-pin nets in the given routing box
n	n^{th} multi-pin net
s^n	A source of n
D^n	Set of sinks of n
d_m^n	m^{th} sink of n
f_m^n	A two-pin subnet connecting s^n and d_m^n, i.e., a commodity
v^n	0-1 indicator if v is used for n
$e_{v,u}^n$	0-1 indicator if $e_{v,u}$ is used for n
$f_m^n(v,u)$	0-1 indicator if $e_{v,u}$ is used for commodity f_m^n
$m_{v,u}$	0-1 indicator if there is a metal segment on $e_{v,u}$
$C_m^n(v,u)$	Capacity variable for $e_{v,u}$ of commodity f_m^n
$g_{d,v}$	0-1 indicator if v forms d-side EOL of a metal segment

linear programming are restricted to be integer, such problems are called ILP (integer linear programming) as expressed in (1.1).

$$\begin{aligned} &\textbf{Minimize} \quad c^T x \\ &\textbf{Subject to} \quad Gx \leq h \\ &\qquad\qquad x \in \mathbb{Z} \end{aligned} \tag{1.1}$$

Since the search space of integer variable's domain is discrete, the problem of ILP is a non-convex problem. Therefore, exhaustive combinatorial searching is required to achieve the optimal solution. The popular method for solving ILP is "Branch and Bound". After quickly finding optimal solution (but, non-integer) of LP, "relaxation" of ILP without integer constraints, several efficient searching algorithms (including "Branch and Bound") are executed to find minimized integer solution with respect to object function in state-of-the-art ILP solvers [6, 7].

1.1.1 Binarized/Undirected Multi-Commodity Network Flow theory

By virtue of its ability to obtain optimal solutions, ILP-based optimization has been widely applied to physical design. Previous works successfully describe ILP-based detailed routing formulation for standard cell synthesis [8] and detailed routers [9, 10] based on popular multi-commodity network flow theory (MCNF) [11]. Recently, Kang *et al.* [5, 12] proposed a binarized MCNF formulation to define source-sink connectivity of each net at a per-commodity granularity (so-called *SAT-friendly ILP*). Inspired by [5, 12], we propose further refined MCNF without flow direction as shown in (1.2), resulting in fully binarized formulation (i.e., binary integer linear programming).

$$f_m^n(v,u) = f_m^n(u,v), \quad \textit{No Direction} \tag{1.2}$$

Since ILP formulation has been fully binarized, we are able to precisely convert/compare ILP formulation to SAT formulation. This is presented in Chapter 2.

1.2 Boolean Satisfiability (SAT) for Detailed Routing

The Boolean Satisfiability (SAT) technique is one of the most successful automated logical reasoning methods in computer science including artificial intelligence (AI). A wide range of practical and challenging questions have been encoded to SAT formulas. A proposition logic formula of SAT consists of variables, AND (i.e., conjunction), OR (i.e., disjunction), and NOT (i.e, negation). The SAT problem is, given a proposition logic formula, to find a variable assignment to make formula evaluates to 1 (i.e., Satisfiable), or prove that no such assignment exists (i.e., Unsatisfiable). The CNF (i.e., product of sum), a conjunction of clauses[2], is basic input format for most of state-of-the-art SAT solvers.

By virtue of SAT's fast reasoning, SAT techniques are widely applied in physical design. However, adopting SAT has been limited to standard cell routing [13], escape routing [14], and special-purpose ICs (e.g., FPGA [15, 16] and cross-referencing biochip [17]) due to the high complexity of expressing "countable" design metrics and constraints in DR problems incorporating complex conditional design rules. Recently, Kang *et al.* [5, 12] proposed a SAT-based routability analysis framework with design rule-correct assessment, resulting in fast routability analysis for early "go/no-go" decision. Although the SAT-based routability analysis is feasible in physical design, further work is still open in terms of practical scalability. Furthermore, once we identify the routability, the diagnosis remains an open problem for physical designers.

In this dissertation, we propose a light-weight CNF representation of routability analysis framework to achieve better practicality (Chapter 2). Also, we propose a framework precisely diagnoses explicit reasons for DRVs in the form of human-interpretable explanations while specifying conflicting design rules with a physical location. It means that our routability diagnosis framework is a a special version of SAT solver for describing DRVs expertly. (Chapter 3). The rest of this section, we introduce several SAT techniques for the complexity refinement and the

[2] A *literal* is either a variable (i.e., *positive literal*) or the negation of a variable (i.e., *negative literal*). A *clause* is a disjunction of literals (or a single literal).

diagnosis.

1.2.1 SAT Encodings for Boolean cardinality constraint

Since we utilize MCNF as the foundation of our routing formulation, the majority of our formulation is exclusiveness constraint such as vertex and edge exclusiveness. Furthermore, we adopt geometric variable-based exclusiveness to ensure conditional design rules [5, 12]. In SAT problem, encoding for these exclusiveness (i.e., Boolean cardinality) constraints is non-trivial problem in terms of an efficient CNF representation. For instance, vertex indicator v^n in our formulation is restricted by vertex exclusiveness as shown in (1.3).

$$\sum_{n \in N} v^n \leq 1, \quad \forall v \in V,\ v \neq s^n, d_m^n \tag{1.3}$$

This at-most-1 (i.e., "**AM1**") constraint can be encoded to CNF representation using *binomial* encoding[3] as expressed in (1.4).

$$\bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} \neg\left(v^i \wedge v^j\right) \equiv \bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} \left(\neg v^i \vee \neg v^j\right) \tag{1.4}$$

Although *binomial* encoding for **AM1** is straightforward, it is not efficient in terms of the CNF formula because $\binom{n}{2}$ clauses are required to represent all possible combinations of two variables. Accordingly, intensive studies for the efficient CNF representation of **AM1** are reported [18, 19]. In our routing analysis, we apply *commander* encoding [20] which has the most refined complexity (in our problem) in terms of clause number. In a running example set $X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9\}$, the *binomial* encoding requires $\binom{9}{2} = 36$ clauses (i.e., the

[3]This encoding scheme has several different names such as *binomial*, *pairwise* and *naïve*.

clause complexity is $O(n^2)\big)$ in CNF as described in (1.4). Meanwhile, the *commander* encoding with three disjoint subsets $G_1 = \{x_1, x_2, x_3\}$, $G_2 = \{x_4, x_5, x_6\}$, and $G_3 = \{x_7, x_8, x_9\}$, and three commander variables c_1, c_2, and c_3 requires three exactly-1[4] (i.e., "**E1**") for each group and one **AM1** for commander variables $\big($i.e., total 24 clauses and the clause complexity is $O(n)\big)$ as described in (1.5) and (1.6).

$$\bigwedge_{i=1}^{3} \mathbf{E1}\big(\{\neg c_i\}, G_i\big) = \bigwedge_{i=1}^{3} \Big(\mathbf{AM1}\big(\{\neg c_i\}, G_i\big) \wedge \mathbf{AL1}\big(\{\neg c_i\}, G_i\big)\Big) : \ 21 \ \textit{clauses} \tag{1.5}$$

$$\textit{binomial} \ \mathbf{AM1}(c_1, c_2, c_3) : \ 3 \ \textit{clauses} \tag{1.6}$$

This is presented in Chapter 2.

1.2.2 Boolean Constraint Propagation (BCP)

BCP (Boolean Constraint Propagation) , also called unit propagation, is the basic procedure of most of SAT solver to meet the conflict and reduce clause complexity [21]. Using unit clauses (i.e., clauses that have only one literal.), BCP is executed, resulting in the simplified CNF representation. If a set of clauses contains the unit clause a, the other clauses are simplified by two following rules.

- Every clause containing a literal is removed (i.e., the clause is satisfied regardless of other literals.).
- In every clause, $\neg a$ literal is deleted (i.e., $\neg a$ does not contribute the satisfiability of the clause containing $\neg a$.).

[4]Exactly-1 (**E1**) is the conjunction of at-most-1 (**AM1**) and at-least-1 (**AL1**).

BCP procedure is recursively executed until the repeatedly simplified set of clauses has no more unit clauses. A running example illustrates this BCP procedure in Figure 1.4.

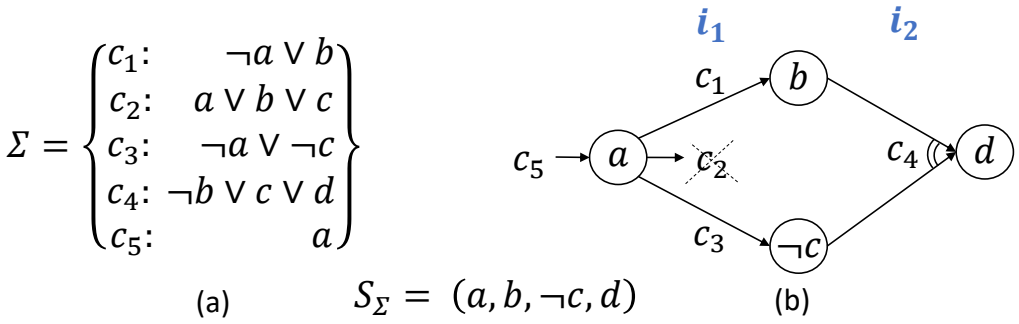


Figure 1.4: An example of (a) SAT formula ∑, and (b) the associated implication graph through BCP.

In the running example, the SAT formula ∑ consists of 5 clauses including one unit clause (i.e., c_5) as shown in Figure 1.4(a). At the first iteration i_1 of BCP, two unit clauses (i.e., c_1 and c_3) are generated and one clause (i.e., c_2) is removed from the formula. In the next iteration i_2 of BCP, one unit clause (i.e., c_4) is generated. As a result, the satisfiable variable assignment S_Σ $\big($i.e., $(a, b, \neg c, d)\big)$ is found. Furthermore, the implications between variables is identified through BCP as depicted in Figure 1.4(b).

In this dissertation, we utilize the BCP procedure as a built-in SAT pre-processor in our routability analysis framework to refine the complexity of CNF representation (Chapter 2). Furthermore, we utilize DAG (directed acyclic graph)-based implication graphs for identifying the root cause of the conflict, while our routability diagnosis framework is specifying conflicting design rules (Chapter 3).

1.2.3 Minimal Unsatisfiable Subset (MUS)

A MUS (Minimal Unsatisfiable Subset) is a clause subset of an infeasible constraint formula which is both unsatisfiable and not able to make smaller without becoming satisfiable [22, 23]. In other words, an unsatisfiable subset of infeasible formula is MUS if all its proper subsets

are satisfiable. For example, a given infeasible formula $\sum_U$ as shown in Figure 1.5(a), possible MUSes are $\{c_2, c_4\}$ and $\{c_1, c_2, c_3\}$ as extracted in Figure 1.5(b).

$$\Sigma_U = \begin{cases} c_1: & a \\ c_2: & b \\ c_3: & \neg a \vee \neg b \\ c_4: & \neg b \\ c_5: & a \vee b \vee c \\ c_6: & a \vee c \end{cases}$$

(a)

MUSes(Σ_U)	c_1	c_2	c_3	c_4	c_5	c_6
$\{c_2, c_4\}$		X		X		
$\{c_1, c_2, c_3\}$	X	X	X			

(b)

Figure 1.5: An example of (a) infeasible SAT formula $\sum_U$, and (b) the corresponding MUSes.

In our routability diagnosis framework, this MUS significantly contributes the diagnosis time reduction by replacing the diagnosis target with the very small number of original clauses. This is presented in Chapter 3.

1.3 Satisfiability modulo theories (SMT) for Standard Cell Layout

Based the fast reasoning ability of SAT, SMT (Satisfiability modulo theories) methodologies empowers us to represent more expressive language containing non-Boolean variables (e.g., integer, bit-vector, etc.) and predicate symbols [24]. With ILP and SAT, SMT is also a successful logical reasoning technique in various applications of computer science such as microfluidic platform [25], cyber-physical system [26, 27], vehicle routing [28], , analog placement [29]. and floorplanning of VLSI physical design [30]. Recently, several state-of-the-art SMT solvers with the optimization methodology (so-called "OMT") are released [31, 32].

In this dissertation, we utilize SMT to represent the standard cell layout design problem because SMT-based methodologies support a much richer modeling language than SAT or ILP

formulas. For example, logical constraints (e.g., "if-then-else" for the "Either-Or" constraint) are able to be easily implemented by "ITE" keyword, meanwhile ILP formula requires additional auxiliary variables for logical constraints. Furthermore, an state-of-the-art SMT solver [31] include built-in Boolean cardinality functions such as at-most k (i.e., "**AMk**") and at-least k (i.e., "**ALk**") as introduced in Section 1.2.1. This is represented in Chapter 4.

For the generation of a whole set of a standard cell library, we propose various tactics to mitigate the intrinsic scalability limitation of the exact solving. The rest of this section introduces one of our approaches to prune search space based on solution symmetry. This scheme is utilized in Chapter 5.

1.3.1 Breaking symmetry predicates

To extend the scalability in a combinatorial solving similar to our problem (i.e, the CSP-based exact solving for the optimization), many previous works suggest breaking symmetry predicates as additional constraints to prune the symmetry search space [33, 34, 35]. If the formula preserves the theory (i.e., satisfiability and optimization results in our problem) in spite of permutation of variable assignment, its variable permutation denotes the symmetry in the search space. And then, we never want to visit more than one of them. For this symmetry, we can "break" the symmetry by adding a constraint with logical implication between variables (e.g., $a \implies b$ for interchangeable variables a and b). This logical implication construct a lexicographic order on the set of assignments (i.e., $a \leq b$ for interchangeable variables a and b). Therefore, the search space, not following the constructed order (i.e., $a > b$), will be pruned during the solving. In our standard cell layout generation, the solutions of the transistor placement have a few symmetries. This is presented in Chapter 5.

1.4 Preliminaries of Framework Configurations

As manufacturing technology nodes continuously evolve into sub-10*nm*, routing between standard cell in DR stage has become more challenging due to scaling barriers such as higher pin density and resolution limitations of optical lithography [9, 36, 37]. Although multi-patterning techniques such as LELE (litho-etch-litho-etch), SADP and SAQP (self-aligned double and quadruple patterning) enable the manufacturing of sub-10*nm* technology nodes [38, 39, 40], these approaches induce more complex conditional design rules and restricts routing wires to grid-based (i.e., on-grid) unidirectional routing track due to the process resolution. Therefore, the grid based technology fits the ILP, SAT and SMT methodology (i.e., reasoning techniques for discrete models) proposed in this dissertation. With this in mind, the rest of this section introduce our framework configurations related to routing graph definition, supernode for source and sink connectivity, and boolean cardinality constraint for conditional design rules.

1.4.1 On-grid Unidirectional Routing Graph: G(V, E)

We use a multiple metal layered routing graph $G = (V, E)$ to represent available routing resources (e.g., horizontal and vertical tracks on multiple metal layers, inter-layer vias) and routing between sources and sinks. Each vertex v is mapped to coordinates (x_v, y_v, z_v), where x, y, and z are induced from horizontal routing tracks, vertical routing tracks, and metal layer, respectively as depicted in Figure 1.6. Each edge $e_{v,u}$ between vertex v and vertex u represents flow with capacity one, including inter-layer vias.

1.4.2 Supernode

For the consistency of our MCNF formulation, we adopt supernode to group multiple feasible pin locations as [10, 5, 12]. Figure 1.7 illustrates supernodes in G. A supernode for a pin on $M1$ (i.e., inner pin) is connected to vertices covering the pin (red circles on $PIN1$ and $PIN2$).

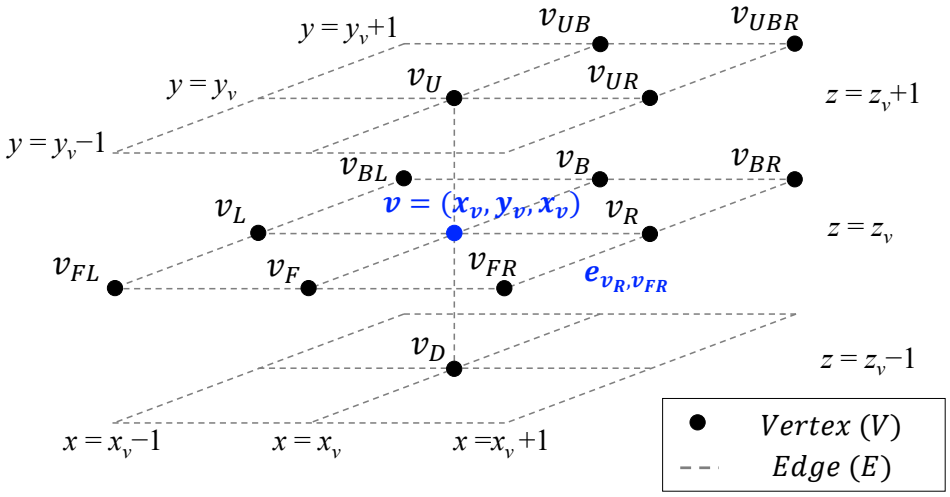


Figure 1.6: An example of grid-based routing graph, $G(V,E)$.

A supernode for a pin connected to vertices at the boundary of G (i.e., outer pin) is depicted in Figure 1.7 (green squares). Using supernode composition, each commodity consists of one source and one sink as supernode, respectively. In particular, our supernode for outer pins (i.e., Super Outer Node S_o) abstracts all supernodes of outer pins so that our framework has only one set of variables for the supernode (i.e, shared supernode). Thus, the complexity of exclusiveness-related constraint is reduced compared to [5, 12]. Note that we denote the pin on $M1$ by Super Inner Node S_i to distinguish from S_o.

1.4.3 Geometric variable for conditional design rules

We adopt geometric variable (GV) $g_{d,v}$ determined by the EOL of a metal segment as shown in the Constraint (1.7).

$$\begin{aligned} g_{L,v} &= \neg m_{v_L,v} \wedge m_{v,v_R}, \\ g_{R,v} &= m_{v_L,v} \wedge \neg m_{v,v_R}, \end{aligned} \qquad \forall v \in V_2 \tag{1.7}$$

Figure 1.8 shows an example to determine $g_{d,v}$. Since the left-/back- and right-/front-directional

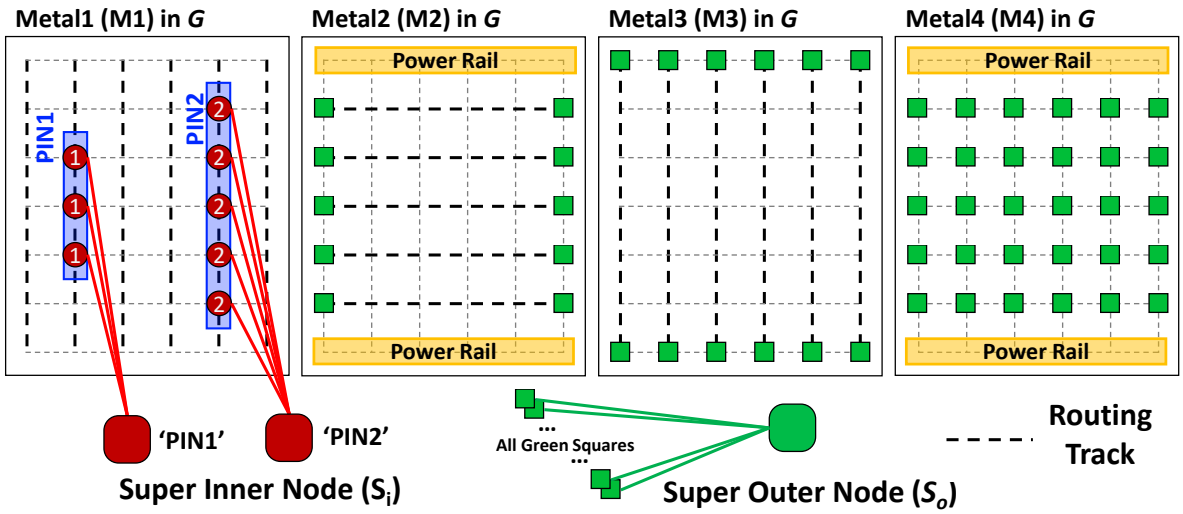


Figure 1.7: Supernodes. $PIN1$ and $PIN2$ respectively cover three and five vertices on $M1$ layer (i.e., inner pin). Outer pins are connected to boundary vertices of G through Super Outer Node (S_o).

EOLs (depicted in red and blue bars) are respectively located at $(1,v)/(v,2)$ and $(2,v)/(v,1)$, $g_{L,(1,v)}/g_{B,(v,2)} = 1$ and $g_{R,(v,2)}/g_{F,(v,1)} = 1$. We convert logical Constraint (1.7) into linear Constraint (1.8) 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 [9]. The front- and back-directional GV are derived by changing L and R to F and B, respectively.

$$\begin{aligned} g_{L,v} \leq 1 - m_{v_L,v} \;;\; g_{L,v} \leq m_{v,v_R} \;;\; g_{L,v} \geq m_{v,v_R} - m_{v_L,v} \\ g_{R,v} \leq m_{v_L,v} \;;\; g_{R,v} \leq 1 - m_{v,v_R} \;;\; g_{R,v} \geq m_{v_L,v} - m_{v,v_R} \end{aligned} \tag{1.8}$$

With GVs, we are able to represent all conditional design rules as the **AM1** constraint, which is the simplified boolean cardinality constraints. With this simple representation for conditional design rules, we can easily add new kinds of design rules into our framework without significant burdens in terms of complexity.

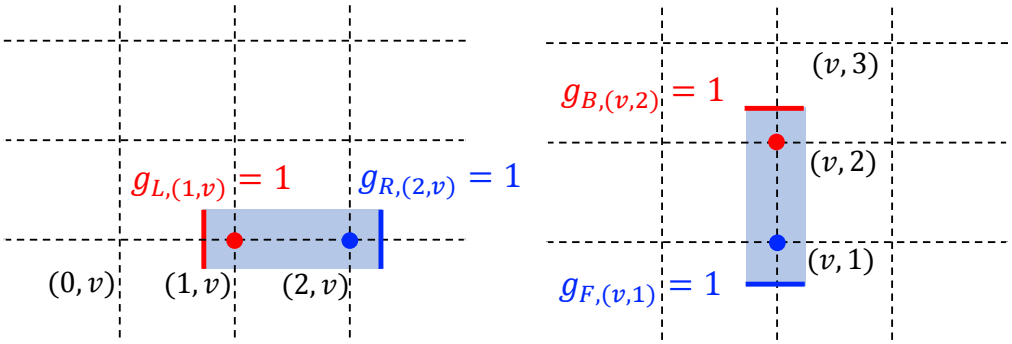


Figure 1.8: An example to determine geometric variable $g_{d,v}$.

Chapter 2

Detailed Routing: Routability Analysis and Optimization Framework

2.1 Introduction

As manufacturing technology nodes continuously evolve into sub-10*nm*, detailed routing has become more challenging [41]. One of the major difficulties is exposed by the resolution limitations coming from the diffraction limit of optical lithography [9, 36, 37]. Although recent multi-patterning techniques such as LELE (litho-etch-litho-etch), SADP and SAQP (self-aligned double and quadruple patterning) enable the delivery of sub-10*nm* technology nodes to foundries [38, 39, 40], multi-patterning approaches induce more 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, the single-point/small-steps elimination, etc.) for manufacturability. In addition to complex design rules, routability becomes a critical bottleneck due to fewer number of routing tracks, higher pin density, and smaller pin geometry [37, 42]. Furthermore, multi-pattern-based DFM (design for manufacturability) restricts routing wires to grid-based unidirectional routing track due to the process resolution in sub-10*nm* technologies [43]. The grid based technology fits the ILP and SAT methodology proposed in this paper.

2.1.1 Related Works

Recently, routability (including pin-accessibility) optimization under DFM constraints has been intensively studied in physical design (PD) as shown in Figure 2.1. Pin accessibility- and BEOL (back end of line)-aware cell layout optimizations [8, 37, 13, 44], and three-dimensional (3-D) monolithic standard cells to improve pin accessibility [45] are introduced. Previous ISPD-2014 [46] and ISPD-2015 [47] contests are dedicated to detailed routing-driven placement. Placement migrations to mitigate local routing congestion [48, 49] and pin accessibility-aware detailed placement refinements [50, 36] are proposed. For detailed routing, an algorithm for dense pin clusters [51] and a pin-access planning framework comprehending SADP [42] are presented.

Despite the previous studies, there still exist pin accessibility challenges 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 work in [9] describes ILP-based concurrent detailed routing formulation based on directed multi-commodity flow, which can comprehend conditional design rules for advanced technologies. ILP-based approaches are computationally-expensive and unaffordable for quick routability analysis, although Jia *et al.* [10] propose several strategies for complexity reduction.

SAT for PD has been adopted for standard cell routing [13], escape routing [14], and special-purpose ICs such as FPGA [15, 16] and cross-referencing biochips [17]. However, work that incorporates conditional design rules is limited due to the high complexity of expressing

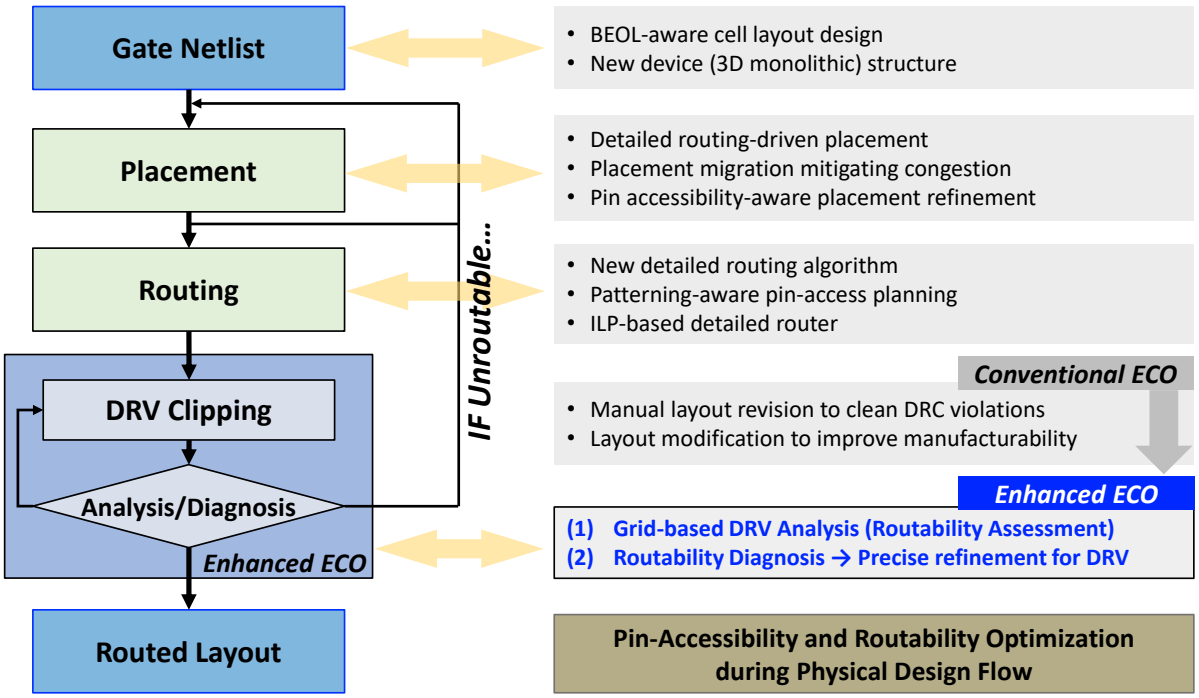


Figure 2.1: Efforts to secure the pin accessibility during the PD procedure. Failure to produce routable (or routed) design indicates loop-back of PD procedure, causing undesired additional ECO (Engineering Change Order) cost. Our SAT-based routability analysis enables a fast and precise DRV assessment, and routability diagnosis minimizes the ECO cost with precise suggestion for DRV refinements.

"countable" design metrics and constraints into a CNF (conjunctive normal form) representation.

Recently, Kang *et al.* [5] proposed a concurrent routability analysis framework with design rule-correct assessment on top of grid-based routing graph and SAT formulation (i.e., Routability Analysis of (1) in Figure 2.1), resulting in fast routability analysis for early "go/no-go" decision in the ECO (Engineering Change Order) stage. However, even if we achieve the SAT feasibility in PD, further work is still open in terms of scalability (e.g., complexity analysis) and practicality (e.g., benchmark and multi-pattern-aware design rules).

2.1.2 Our Contributions

In this chapter, we propose two novel frameworks which enable routability analysis and diagnosis with design rule-correct routing solution through a light-weight grid-based SAT formulation. Our contributions are as follows.

- We propose a fast and precise framework for routability analysis in the ECO stage, which leverages a novel SAT formulation and comprehends net structure (i.e., source-sink connectivity) in a concurrent manner.
- Our framework covers a wide variety of conditional design rules for DFM, e.g., SADP-aware conditional design rules, resulting in DRC (design rule checking)-clean layout.
- Our framework utilizes a grid-based undirected multi-commodity flow foundation in both ILP- and SAT-formulation. The ILP formulation is used for high-level encoding of design rules, thus the formulas are concise and comprehensible for the users. The SAT formulation is automatically converted from ILP with the minimized number of literals and clauses. Furthermore, our framework applys supernode simplification scheme for more refined representations.
- We demonstrate that our framework performs routability analysis within 0.03% of ILP (i.e., conventional approaches) runtime on average.

- We demonstrate that our framework's scalability using a state-of-the-art practical benchmark (ISPD2019) routed by the commercial detailed routing tool.

The remaining sections are organized as follows. Section 4.2 introduces framework configuration. Section 2.3 describes our constraint formulation including ILP-, SAT-, and ILP-to-SAT conversion scheme. Section 2.4 presenta our routabiliy analysis framework procedures. Section 2.5 discusses our experimental setup/results. Section 2.6 concludes the paper.

2.2 Routability Analysis Framework Preliminary

Based on Section 1.4, this section introduces our assumptions for routing graph of detailed routing (Section 2.2.1), and formulation configuration (Section 2.2.2).

2.2.1 Grid-based Routing Graph: G(V, E)

As Section 1.4.1 mentioned, our framework adopt on-grid and unidirectional routing graph $G(V,E)$.

No-$M1$ Routing. Since the pin shapes for each standard cell are given in $M1$ layer, we prevent the in-layer routing of $M1$ which means that we only allow to access the pins of standard cell through the vias.

Power Rail. Since the top- and bottom-most routing tracks for each placement row in $M2$ and $M4$ are reserved for power rails, we set all 0-1 indicators on corresponding edges as 0 (i.e., resevered track for power rail (yellow box in Figure 1.7)).

2.2.2 Configuration of Constraint Formulation

Our formulation consists of two subgroups, *flow formulation* and *design rule formulation* as shown in Figure 2.2. The flow formulation based on the multi-commodity flow formulation

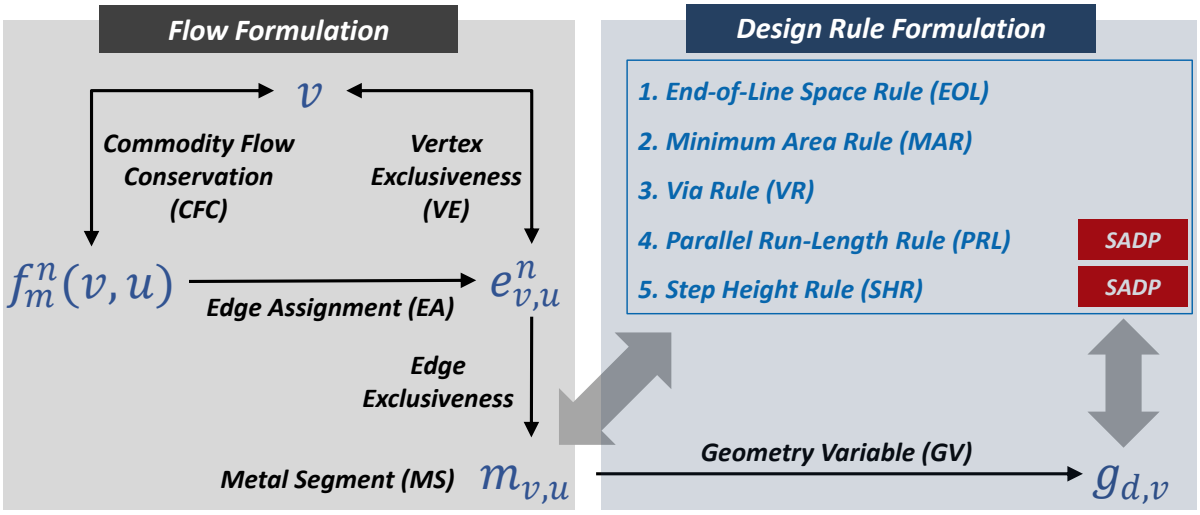


Figure 2.2: Configuration of constraint formulation.

(e.g., commodity flow conservation, vertex exclusiveness, edge assignment, and edge exclusiveness by adopting metal segment) secures routing path between the source and sink for each commodity without heuristic modeling. The design rule formulation restricts design rule violated routing path through the at-most-one (i.e., **AM1**) constraint of metal segments and geometric variables. Our framework implements various conditional design rules including multi-pattern-aware design rules (SADP) such as parallel run-length (PRL) rule and step height rule (SHR). Through the minimization of the number of metal segments, we provide the optimal routing solutions with DFM-aware routability.

2.3 Constraints Formulation

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 (i.e., per each sink) [52]. To facilitate our ILP-to-SAT conversion, all constants in our ILP formulation have either 0 or 1 (so-called *SAT-friendly ILP*). From ILP-to-SAT conversion, we successfully adopt a SAT formulation in PD by virtue of 0-1 manner ILP

formulation.

2.3.1 SAT-friendly ILP formulation

We formulate our ILP-based routing formulation on the basis of the multi-commodity flow theory [53, 5]. The objective of our suggested ILP formulation is the total sum of weighted metal lengths as shown in Equation (2.1). We define inter-layer via cost as 4 and metal cost on the same metal layer as 1.

$$\textbf{Minimize}: \quad Cost = \sum_{e_{v,u} \in E} c_{v,u} \times m_{v,u} \tag{2.1}$$

In the flow formulations, we adopt a multi-commodity flow foundation to represent routing flows for a given layout. The flows are defined as an undirected graph to reduce the solution space as shown in (1.2).

Commodity Flow Conservation (CFC). The CFC on each vertex and its connected edges ensure source-to-sink connectivity (i.e., from a supernode to another supernode) of each commodity flow, as shown in Constraint (2.2).

$$\sum_{u \in a(v)} f_m^n(v,u) = \begin{cases} 1, & \text{if } v = s^n, d_m^n \\ 2p, p = \{0,1\}, & \text{otherwise} \end{cases} \tag{2.2}$$
$$\forall v \in V,\ \forall n \in N,\ \forall d_m^n \in D^n$$

As described in Section 1.4.2, we adopt supernode to represent each pin. For pin (i.e., supernode $v = s^n, d_m^n$), CFC allows only a single commodity (i.e., **AM1** constraint) as shown in Figure 2.3. For any non-pin (grid) vertex (i.e., $v \neq s^n, d_m^n$), the flow indicator $f_m^n(v,u)$ is either 0 or 2,[1] depicted in Figure 2.4. If there are no flow passes for the vertex v, all flow indicators $f_m^n(v,u)$ are set as

[1] We use an auxiliary variable p to maintain the 0-1 manner of our formulation as Constraint (4.7).

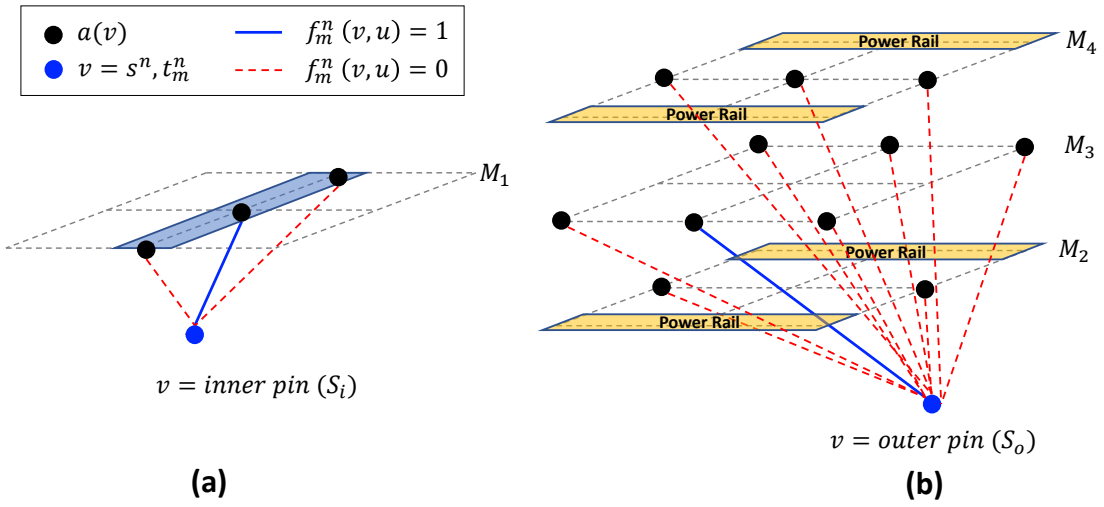


Figure 2.3: An example of CFC constraint for pins (i.e., $v = s^n, d_m^n$). (a) CFC of virtual edges for inner pin. (b) CFC of virtual edges for outer pin.

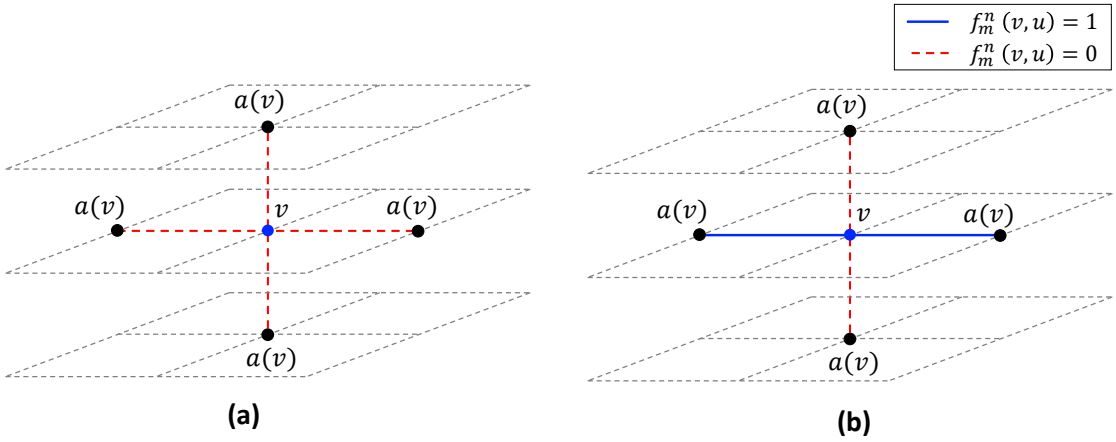


Figure 2.4: An example of CFC constraint for grid (a) no flow passes, (b) the flow of commodity m is connected between a pair of edges.

0 as shown in Figure 2.4(a). Otherwise, the flow of commodity m must use a pair of connected edges as shown in Figure 2.4(b).

Vertex Exclusiveness (VE). A vertex v should be used by only a single net (except S_o). Constraint (2.3) ensures that there are no intersecting nets on any vertices.

$$\sum_{n \in N} v^n \begin{cases} = 1, & \text{if } v = s^n, d_m^n \\ \leq 1, & \text{otherwise} \end{cases}, \quad \forall v \in V,\ v \neq S_o \tag{2.3}$$

When $v = s^n, d_m^n$, the only one edge into vertex v must be used as represented in Constraint (2.4) (i.e., exactly-1 (**E1**)).

$$v^n = \sum_{u \in a(v)} e_{v,u}^n, \quad v = s^n, d_m^n, \quad \forall n \in N \tag{2.4}$$

When $v \neq s^n, d_m^n$, we allow the multiple uses of edges with respect to vertex v for a certain net as shown in Constraint (2.5) (i.e., logically ORing all adjacent edges of v) to preserve multi-commodity flows in a net.

$$v^n = \bigvee_{u \in a(v)} e_{v,u}^n, \quad v \neq s^n, d_m^n, \quad \forall n \in N \tag{2.5}$$

Edge Assignment (EA). To obtain a Steiner tree of net n, we determine the edge indicator $e_{v,u}^n$ by overlapping each commodity flow belonging to the net n. As shown in Constraint (2.6), $e_{v,u}^n$ can be either 0 or 1 even the flow indicator $f_m^n(v,u) = 0$ (i.e., logical implication $f_m^n(v,u) \implies e_{v,u}^n$), ensuring the multi-commodity flow of n (Steiner points are naturally obtained).

$$e_{v,u}^n - f_m^n(v,u) \geq 0, \quad \forall e_{v,u} \in E,\ \forall n \in N,\ \forall d_m^n \in D^n \tag{2.6}$$

Metal Segment (and Edge Exclusiveness) (MS). We adopt the metal indicator $m_{v,u}$ to determine whether a metal segment is on $e_{v,u}$ or not. Constraint (2.7) ensures that there are no intersecting nets on any edges. Thus, MS is ensuring the exclusive use of $e_{v,u}$.

$$m_{v,u} = \sum_{n \in N} e_{v,u}^n, \quad \forall e_{v,u} \in E \tag{2.7}$$

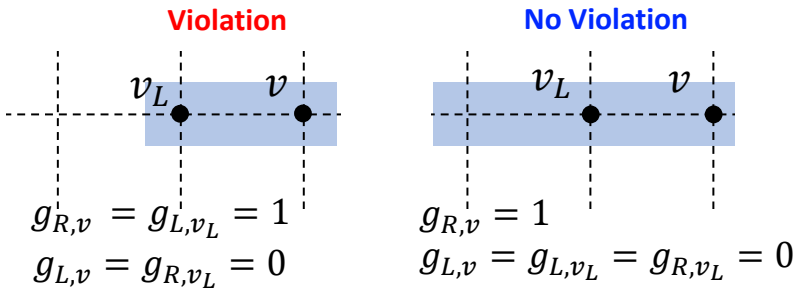


Figure 2.5: An example of the minimum area rule.

For the design rule formulations, our proposed formulation adopts representative conditional design rules such as *Minimum Area Rule*, *EOL Spacing Rule*, and *Via Rule* from [8, 10]. Moreover, our framework includes multi-pattern-aware design rules such as *PRL rule* and *SHR* [54, 40]. Note that all conditional design rules in our framework can be represented by using any integer numbers of the grids (i.e., flexible design-rule functionality).

Minimum Area Rule (MAR). Each disjoint metal segment should be larger than the minimum manufacturable size. Constraint (2.8) ensures that there are no metal segments violating the MAR. Figure 2.5 illustrates the MAR formula if we assume that a metal segment must cover at least three vertices.

$$g_{L,v} + g_{R,v} + g_{L,v_L} + g_{R,v_L} \leq 1, \quad \forall v \in V_2 \tag{2.8}$$

End-of-Line (EOL) Spacing Rule. The distance between each EOL of two metal segments that are coming from opposite directions should be greater than the minimum spacing distance. Constraint (2.9) and Figure 2.6 describe the right-directional EOL when we assume that the minimum distance between any of two opposite EOLs must be larger than L_1 norm (i.e., Manhattan distance) of two vertices in G. The left-, front-, and back-directional EOLs are similarly derived.

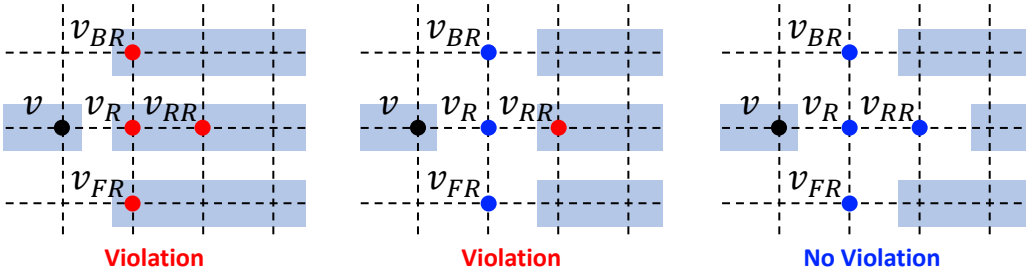


Figure 2.6: An example of the end-of-line (EOL) spacing rule.

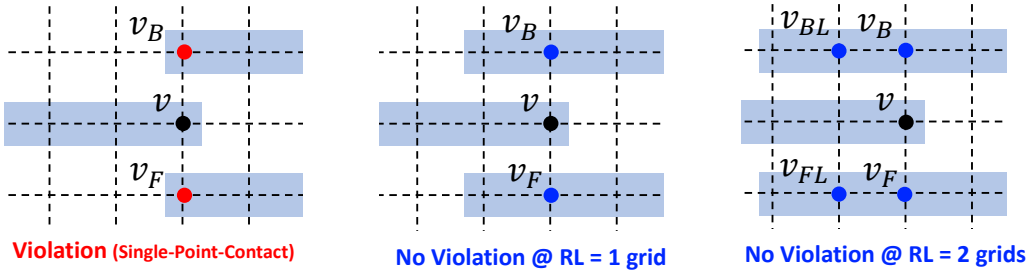


Figure 2.7: An example of the Parallel Run Length (PRL) rule.

$$\begin{aligned} &g_{R,v} + g_{L,v_{FR}} \leq 1; \\ &g_{R,v} + g_{L,v_R} + g_{L,v_{RR}} \leq 1; \qquad \forall v \in V_2 \\ &g_{R,v} + g_{L,v_{BR}} \leq 1; \end{aligned} \tag{2.9}$$

Parallel Run Length (PRL) Rule. PRL rule enforces the avoidance of "single-point-contact" in SADP mask manufacturing [40]. Figure 2.7 and Constraint (2.10) represent PRL rule and the corresponding formulation when run length is 2 grids.

$$\begin{aligned} &g_{R,v} + g_{L,v_B} + g_{L,v_{BL}} \leq 1; \\ &g_{R,v} + g_{L,v_F} + g_{L,v_{FL}} \leq 1; \end{aligned} \qquad \forall v \in V_2 \tag{2.10}$$

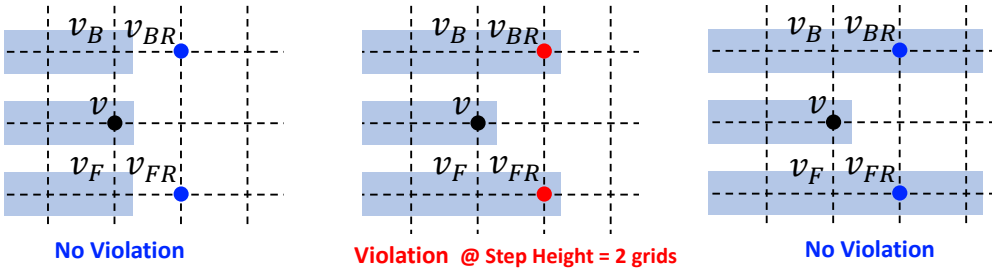


Figure 2.8: An example of Step Height Rule (SHR) @ Step Height = 2 grids.

Step Height Rule (SHR). SHR is for avoiding "the small step" in SADP mask manufacturing [40]. Figure 2.8 and Constraint (2.11) illustrate SHR and the corresponding formulation when the step height is 2 grids.

$$g_{R,v} + g_{R,v_{BR}} \leq 1; g_{R,v} + g_{R,v_{FR}} \leq 1, \quad \forall v \in V_2 \tag{2.11}$$

Via Rules (VR). Logical Expression (2.12) represents VR related to restriction rules of inter-layer via (i.e., via) locations when the minimum distance (i.e., L^2 Norm) between via is $\sqrt{2}$ grids as shown in Figure 2.9(a) (i.e., via-to-via spacing rule). The via placement on top of another via is prohibited as shown in Figure 2.9(b) (i.e., stacked-via regulation).

$$m_{v,v_U} + m_{v_R,v_{UR}} + m_{v_B,v_{UB}} + m_{v_{BR},v_{UBR}} \leq 1, \quad \forall v \in V \tag{2.12}$$

$$m_{v_D,v} + m_{v,v_U} \leq 1, \quad \forall v \in V \tag{2.13}$$

2.3.2 ILP-to-SAT Conversion

We describe our ILP-to-SAT conversion that accelerates runtime of SAT solvers. A general SAT solver takes input in *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

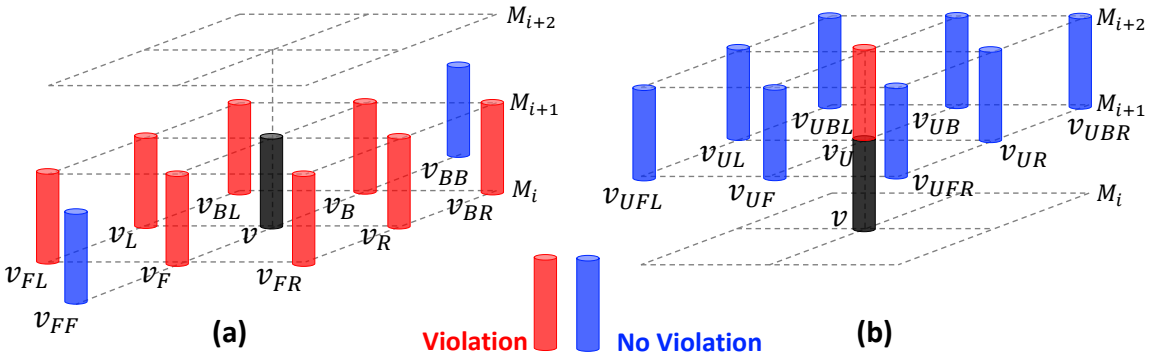


Figure 2.9: VR rule. (a) via-to-via spacing rule. (b) stacked-via regulation

disjunction (OR) of literals.

In terms of SAT conversion (i.e., encoding) from ILP constraints, our formulation utilizes a "muldirect" method [55, 56], a direct encoding method of constraint satisfaction problem. The "muldirect" method allows multiple assignments for the multi-commodity network to represent the detailed routing problem. Moreover, we reduce our SAT's complexity by utilizing the **AM1** constraints' encoding techniques [18] for the **AM1** type constraints and the logic minimizer [57] for the non-**AM1** type constraints as follows.

Constraints Characteristic. To facilitate the ILP-to-SAT conversion, we devise two techniques for our ILP formulation: (i) refining all the variables as a 0-1 indicator and (ii) formulating source-sink connectivity of each net at a per-commodity granularity. To convert an ILP constraint to a set of clauses in a CNF formula, we analyze constraints as shown in Table 2.1. 9 out of 12 constraint types (i.e., CFC for pins, VE for pins, MS, and all conditional design rules) are the class of exclusiveness constraints (i.e., **AM1** or **E1**). Thus, we can apply efficient encoding techniques to reduce the formulation complexity in terms of clause and literal number. In the case of the non-**AM1** constraints, we apply the two-level logic minimizer *Espresso* [57] to get the minimal CNF representation.

SAT Encoding for AM1 constraint. We utilize the encoding technique for the **AM1** type of constraints inspired by [18]. Since [5] has the tool limitation of solving SAT comes from their

Table 2.1: Constraint Characteristics.

Group	Constraint	Description	Type
Flow	CFC for pins	**E1** among $a(v)$	**AM1**
	CFC for grids	0 or 2 edges among $a(v)$	non-**AM1**
	VE for pins	**E1** among $a(v)$ for all n	**AM1**
	VE for grids	**AM1** n use for v	**AM1**
	EA	Logical Imply	non-**AM1**
	MS	Exclusiveness use of edge	**AM1**
Design Rule	GV	Logically AND between MS	non-**AM1**
	MAR/EOL/PRL/SHR	**AM1** GV use	**AM1**
	VR	**AM1** via metal use	**AM1**

"pairwise" encoding $\big($i.e., clause complexity is $O(n^2)\big)$, we apply "commander" encoding [20] which has the most efficient complexity in the number of clauses $\big($i.e., complexity is $O(n)\big)$. Furthermore, we dynamically apply the encoding technique depending on the size of the domain. If the element size of the domain is less than 6, we apply "pairwise" encoding since the formula complexity (i.e., the number of clauses + the number of auxiliary variables) of the "commander" encoding is larger than the complexity of "pairwise" encoding. As a result, all conditional design rules (i.e., MAR, EOL, VR, PRL, and SHR) are formulated by "pairwise" encoding without any auxiliary variables since the element size of the examples, as described in the previous section (Section 2.3.1), is less than 6. On the other hand, CFC for the outer pin, VE and MS constraints are encoded by utilizing the "commander" encoding, resulting in the light-weight SAT formulation.

Logic Simplication for non-AM1 constraint. For non-**AM1** constraints such as CFC, EA, and GV constraint, we exploit our SAT formulation by logic minimizer *Espresso* to obtain the minimal CNF representations (i.e. providing the functionally-equivalent formula) with the fewest number of clauses and literals. By virtue of regularity of the routing graph G, ILP constraints derived by the same ILP formula have 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 structure of ILP constraints and their patterns of the minimal CNF representations. We utilize the preprocessed *Espresso* results as the predetermined minimal CNF representations for each ILP formula. As a result, we generate the minimal CNF formula without recursively calling *Espresso* during our ILP-to-SAT conversion.

2.3.3 SAT Formulation

As discussed above, our proposed SAT formulation can be effectively obtained by converting 0-1 encoded ILP-based constraints via the SAT encoding techniques and logic minimizer.

Flow feasibility **F** provides feasible routing paths for every source-sink connectivity if the given layout is routable. A satisfiable assignment of flow-related grid-based variables determines feasible routing paths for all source-sink connectivity of the nets. For the *Flow feasibility* **F**, each sub-formulation is converted from the ILP-based constraint as follows.

Commodity Flow Conservation (CFC). Feasibility for CFC $\mathbf{F}_C$ (2.14) has a bipartite formulation, either for pin (i.e. $\mathbf{F}_{C_1}$) or for grid (i.e. $\mathbf{F}_{C_2}$), depending on the type of vertices.

$$\mathbf{F}_C(v,n,m) = \begin{cases} \mathbf{F}_{C_1}(v,n,m), & \text{if } v = pin \\ \mathbf{F}_{C_2}(v,n,m), & \text{if } v = grid \end{cases} \tag{2.14}$$

$\mathbf{F}_{C_1}$ (2.15) represents pin-type vertex's CFC feasibility converted from (2.2) by using either "pairwise" (for inner pin) or "commander" (for outer pin) **E1** constraint, for each commodity on a pin-type vertex v with flow indicator $f_m^n(v,u)$.

$$\mathbf{F}_{C_1}(v,n,m) = \mathbf{E1}\big(\{ f_m^n(v,p) \;\big|\; p \in a(v) \}\big), \qquad \forall v \in V,\ \forall n \in N,\ \forall d_m^n \in D^n \tag{2.15}$$

$\mathbf{F}_{C_2}$ (2.16) represents grid-type vertex's CFC feasibility converted from (2.2), which is non-**AM1** type constraint based on the insight that a grid-type vertex must have either none of incoming and outgoing flows connecting a pair of edges.

$$\mathbf{F}_{C_2}(v,n,m) = \bigwedge_{u \in a(v)} \neg f_m^n(v,u) \vee \bigvee_{p,q \in a(v), p \neq q} \left(f_m^n(v,p) \wedge f_m^n(v,q) \wedge \bigwedge_{u \in a(v), u \neq p, u \neq q} \neg f_m^n(v,u) \right), \quad \forall v \in V,\ \forall n \in N,\ \forall d_m^n \in D^n \tag{2.16}$$

Vertex Exclusiveness (VE). Similar to CFC, feasibility for VE $\mathbf{F}_E$ (2.17) has bipartite formulation, either for pin (i.e. $\mathbf{F}_{E_1}$) or for grid (i.e. $\mathbf{F}_{E_2}$), depending on the type of vertices.

$$\mathbf{F}_E(v) = \begin{cases} \mathbf{F}_{E_1}(v), & \text{if } v = pin \\ \mathbf{F}_{E_2}(v), & \text{if } v = grid \end{cases} \tag{2.17}$$

$\mathbf{F}_{E_1}$ (2.18) represents inner-pin-type vertex's VE feasibility converted from (2.3) and (2.4), by using "commander" **E1** constraint.

$$\mathbf{F}_{E_1}(v) = \mathbf{E1}\big(\{ e_{v,u}^n \mid u \in a(v),\ n \in N \}\big),\ \ \forall v \in V,\ v \neq S_o \tag{2.18}$$

$\mathbf{F}_{E_2}$ (2.19) represents grid-type vertex's VE feasibility converted from (2.3) and (2.5), by using "commander" **AM1** constraint of net indicator v^n $\big($i.e., $v^n = \bigvee_{u \in a(v)} e_{v,u}^n$ from (2.5)$\big)$.

$$\mathbf{F}_{E_2}(v) = \mathbf{AM1}\big(\{ v^n \mid n \in N \}\big),\ \ \forall v \in V \tag{2.19}$$

Edge Assignment (EA). Feasibility for EA $\mathbf{F}_{EA}$ (2.20) represents the logical implication $f_m^n(v,u) \implies e_{v,u}^n$ in Boolean formula for each flow indicator $f_m^n(v,u)$.

$$\mathbf{F}_{EA}(e_{v,u},n,m) = \neg f_m^n(v,u) \vee e_{v,u}^n, \qquad \forall e_{v,u} \in E, \forall n \in N, \forall d_m^n \in D^n \tag{2.20}$$

Metal Segment (MS). $\mathbf{F}_M$ (2.21) represents MS feasibility converted from (2.7) by using "commander" **E1** constraint for each edge.

$$\mathbf{F}_M(e_{v,u}) = \mathbf{E1}\big(\{\neg m_{v,u}\} \cup \{e_{v,u}^n \mid n \in N\}\big),\ \forall e_{v,u} \in E \tag{2.21}$$

The *flow feasibility* **F** is represented by a conjunction of each sub-formulation as Expression (2.22).

$$\mathbf{F} = \bigwedge_{v \in V} \left(\mathbf{F}_E(v) \wedge \bigwedge_{n \in N} \bigwedge_{d_m^n \in D^n} \mathbf{F}_C(v,n,m) \right) \wedge \bigwedge_{e_{v,u} \in E} \left(\mathbf{F}_M(e_{v,u}) \wedge \bigwedge_{n \in N} \bigwedge_{d_m^n \in D^n} \mathbf{F}_{EA}\big(e_{v,u},n,m\big) \right) \tag{2.22}$$

Design Rule Feasibility **D** provides the feasible routing path for all source-sink connections corrected by design rules through the conjunction (AND) of **D** and **F** in SAT solving, resulting in design rule corrected routing paths. **D** covers 5 representative conditional design rules (i.e., *MAR*, *EOL Spacing Rule*, *PRL Rule*, *SHR*, and *VR*). Each design rule formulation is converted from the ILP-based constraint as follows.

Geometric Variable (GV). Logical Expressions (2.23-2.25) represent the GV's feasibility $\mathbf{D}_{GV}$. Each geometric-directional feasibility $\mathbf{D}_{GV_d}$ is directly converted from logical Constraint (1.7). The front- and back-directional feasibilities are derived by changing L and R to F and B, respectively. $\mathbf{D}_{GV}$ is the conjunction of each $\mathbf{D}_{GV_L}(v)$, $\mathbf{D}_{GV_R}(v)$, $\mathbf{D}_{GV_F}(v)$, $\mathbf{D}_{GV_B}(v)$ for all vertices.

$$\mathbf{D}_{GV_L}(v) = (g_{L,v} \wedge \neg m_{v_L,v} \wedge m_{v,v_R}) \ \vee \ (\neg g_{L,v} \wedge \neg m_{v,v_R}) \ \vee (\neg g_{L,v} \wedge m_{v_L,v}), \quad \forall v \in V_2 \tag{2.23}$$

$$\mathbf{D}_{GV_R}(v) = (g_{R,v} \wedge m_{v_L,v} \wedge \neg m_{v,v_R}) \ \vee \ (\neg g_{R,v} \wedge \neg m_{v_L,v}) \ \vee (\neg g_{R,v} \wedge m_{v,v_R}), \quad \forall v \in V_2 \tag{2.24}$$

$$\mathbf{D}_{GV} = \bigwedge_{v \in V} \big(\mathbf{D}_{GV_L}(v) \wedge \mathbf{D}_{GV_R}(v) \wedge \mathbf{D}_{GV_F}(v) \wedge \mathbf{D}_{GV_B}(v)\big) \tag{2.25}$$

Based on GV, conditional design rules are represented as follows.

GV-based Conditional Design Rules. Logical Expressions (2.26-2.29) represent each directional feasibility of GV-based conditional design rules (i.e., MAR/EOL/PRL/SHR) using a "pairwise" **AM1** constraint of GVs. The design rule-corrected feasibility for each conditional design rule is represented by the conjunction of each directional feasibility as shown in Expression (2.30-2.33).

$$\mathbf{D}_{M_{LR}}(v) = \mathbf{AM1}\big(g_{L,v}, g_{R,v}, g_{L,v_R}, g_{R,v_R}\big), \qquad \forall v \in V_2 \tag{2.26}$$

$$\mathbf{D}_{E_R}(v) = \mathbf{AM1}(g_{R,v}, g_{L,v_{FR}}) \ \wedge \ \mathbf{AM1}(g_{R,v}, g_{L,v_{BR}}) \ \wedge \mathbf{AM1}(g_{R,v}, g_{L,v_R}, g_{L,v_{RR}}), \forall v \in V_2 \tag{2.27}$$

$$\mathbf{D}_{P_R}(v) = \mathbf{AM1}(g_{R,v}, g_{L,v_B}, g_{L,v_{BL}}) \ \wedge \ \mathbf{AM1}(g_{R,v}, g_{L,v_F}, g_{L,v_{FL}}), \qquad \forall v \in V_2 \tag{2.28}$$

$$\mathbf{D}_{S_R}(v) = \mathbf{AM1}(g_{R,v}, g_{R,v_{BR}}) \wedge \mathbf{AM1}(g_{R,v}, g_{R,v_{FR}}), \forall v \in V_2 \tag{2.29}$$

$$\mathbf{D}_M = \bigwedge_{v \in V} \big(\mathbf{D}_{M_{LR}}(v) \wedge \mathbf{D}_{M_{FB}}(v)\big) \tag{2.30}$$

$$\mathbf{D}_E = \bigwedge_{v \in V} \big(\mathbf{D}_{E_L}(v) \wedge \mathbf{D}_{E_R}(v) \wedge \mathbf{D}_{E_F}(v) \wedge \mathbf{D}_{E_B}(v)\big) \tag{2.31}$$

$$\mathbf{D}_P = \bigwedge_{v \in V} \big(\mathbf{D}_{P_L}(v) \wedge \mathbf{D}_{P_R}(v) \wedge \mathbf{D}_{P_F}(v) \wedge \mathbf{D}_{P_B}(v)\big) \tag{2.32}$$

$$\mathbf{D}_S = \bigwedge_{v \in V} \big(\mathbf{D}_{S_L}(v) \wedge \mathbf{D}_{S_R}(v) \wedge \mathbf{D}_{S_F}(v) \wedge \mathbf{D}_{S_B}(v)\big) \tag{2.33}$$

Via Rule (VR). $\mathbf{D}_V(v)$ (2.34) represents VR feasibility converted from (2.12) and (2.13), by using "pairwise" **AM1** constraints of metal segment $m_{v,u}$.

$$\mathbf{D}_V(v) = \mathbf{AM1}\big(m_{v,v_U}, m_{v_R,v_{UR}}, m_{v_B,v_{UB}}, m_{v_{BR},v_{UBR}}\big) \wedge \mathbf{AM1}\big(m_{v_D,v}, m_{v,v_U}\big), \quad \forall v \in V \tag{2.34}$$

The *design rule correctness* $\mathbf{D}$ (2.35) is represented by the conjunction of each design rule subset .

$$\mathbf{D} = \mathbf{D}_{GV} \wedge \mathbf{D}_M \wedge \mathbf{D}_E \wedge \mathbf{D}_P \wedge \mathbf{D}_S \wedge \bigwedge_{v \in V} \mathbf{D}_V(v) \tag{2.35}$$

The design rule-correct routability $\mathbf{R}$ can be derived by the conjunction of all sub formation of $\mathbf{F}$ and $\mathbf{D}$ as shown in Expression (2.36).

$$\mathbf{R} = \mathbf{F} \wedge \mathbf{D} \tag{2.36}$$

Through the routability analysis (Section 2.4), we evaluate if the expression $\mathbf{R}$ is satisfiable, i.e., there are routable(feasible) solutions with respect to all the given design rules.

2.4 Routability Analysis Framework

This section consists of *(i) Overall flow of Routability Analysis Framework*, *(ii) SAT Complexity Analysis*, and *(iii) An Example of Routability Analysis Result (ILP & SAT)*.

2.4.1 Overall Flow of Routability Analysis Framework

Figure 2.10 illustrates the overall flow of our framework for routability analysis. We create the grid-based switchbox including the information of standard cell's I/O pins and net/commodity reflecting the connectivity between the pins from the detailed placement result [58]. In ILP formulation, we generate boolean-manner linear constraints with respect to each vertex or edge to find the optimal routing path of each source-sink connectivity (i.e., per each commodity) as

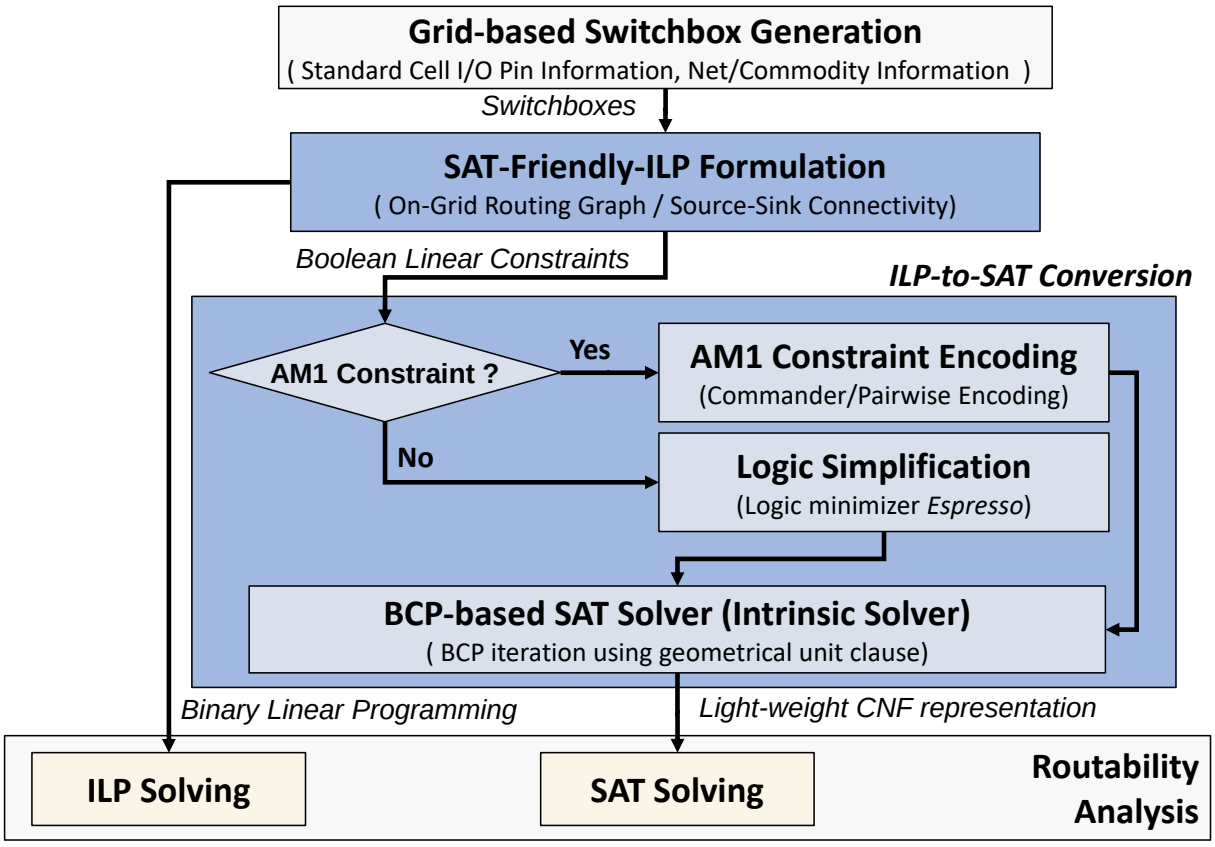


Figure 2.10: Overall flow of our routability analysis framework

described in Section 2.3.1. In ILP-to-SAT conversion phase, we refine each constraint depending on the type of the constraint (i.e., **AM1** type or non-**AM1** type). If the constraint is **AM1** type, we apply the clause-efficient "commander" encoding technique. Otherwise, we utilize the logic minimizer *Espresso* [57] to obtain the efficient CNF representation. Before executing the SAT solver, we implement an intrinsic BCP (Boolean Constraint Propagation)-based SAT solver using geometrical initial assignment to produce further light-weight representation.

Using the resulting formulations, our framework achieves (i) the design rule-correct optimal routing solution through ILP-based detailed routing procedure; and (ii) a design rule-correct routing feasibility assessment through the SAT-based routability analysis procedure.

2.4.2 Complexity Analysis for SAT-based Framework

Table 2.2 presents the complexity of our SAT formulation. Since our formulation supports multi-commodity flows on top of the grid-based routing graph, the complexity is significantly related to the number of the grid (i.e., vertex v or edge $e_{v,u}$) and commodity d_m^n as explained in Section 2.3.3. Equation (2.37) represents the formulas for the number of vertices and edges in a given switchbox. The layout size (i.e., $X \cdot Y$) determines the cardinalities of sets V and E.

$$|V| = 4 \cdot X \cdot Y, \quad |E| \approx 7 \cdot X \cdot Y \tag{2.37}$$

Complexity of Flow Formulation. As shown in Table 2.1, 4 out of 6 constraint types for the flow formulation are **AM1** type and the dominant portion of formulation. Thus, the complexity of flow formulation is linearly increased with respect to each argument (i.e., $|V|$, $|E|$, $|N|$, and $|T|$) because we apply the "commander" encoding for **AM1** type constraints.

In the case of CFC constraint for grids, the number of clauses/literals are proportional to $|V| \cdot |T|$ because our formulation defines grid-related CFC constraints per vertex v, and per

Table 2.2: Analysis for complexity of SAT formulation. In the table, $|N|$ = #Nets, $|T|$ = #Commodities, $|S_o|$ = #OuterPins, $|S_i|$ = #InnerPins, $|P|$ = #Pins, $|D|$ = #Design_rules. $|P| = |T| + |N|$.

Group	Constraint	#Clauses	#Literals	Complexity
Flow	CFC for grids	$\approx 96 \cdot X \cdot Y \cdot \lvert T \rvert$	$\approx 224 \cdot X \cdot Y \cdot \lvert T \rvert$	$O\big(X \cdot Y \cdot \lvert T \rvert\big)$
	CFC for pins	$\approx 14 \cdot X \cdot Y \cdot \lvert S_o \rvert$	$\approx 28 \cdot X \cdot Y \cdot \lvert S_o \rvert$	$O\big(X \cdot Y \cdot \lvert S_o \rvert\big)$
	VE for grids	$\approx 30 \cdot X \cdot Y \cdot \lvert N \rvert$	$\approx 60 \cdot X \cdot Y \cdot \lvert N \rvert$	$O\big(X \cdot Y \cdot \lvert N \rvert\big)$
	VE for pins	$\approx 10 \cdot \lvert S_i \rvert$	$\approx 20 \cdot \lvert S_i \rvert$	$O\big(\lvert S_i \rvert\big)$
	EA	$7 \cdot X \cdot Y \cdot \lvert T \rvert$	$14 \cdot X \cdot Y \cdot \lvert T \rvert$	$O\big(X \cdot X \cdot \lvert T \rvert\big)$
	MS	$\approx \frac{49}{2} \cdot X \cdot Y \cdot \lvert N \rvert$	$\approx 49 \cdot X \cdot Y \cdot \lvert N \rvert$	$O\big(X \cdot Y \cdot \lvert N \rvert\big)$
Flow Complexity		$\approx 103 \cdot X \cdot Y \cdot \lvert P \rvert$	$\approx 238 \cdot X \cdot Y \cdot \lvert P \rvert$	$O\big(X \cdot Y \cdot \lvert P \rvert\big)$
Design Rule	GV	$\approx 12 \cdot X \cdot Y$	$\approx 28 \cdot X \cdot Y$	$O\big(X \cdot Y\big)$
	MAR	$\approx 12 \cdot X \cdot Y$	$\approx 24 \cdot X \cdot Y$	
	EOL	$\approx 20 \cdot X \cdot Y$	$\approx 40 \cdot X \cdot Y$	
	PRL	$\approx 24 \cdot X \cdot Y$	$\approx 48 \cdot X \cdot Y$	
	SHR	$\approx 8 \cdot X \cdot Y$	$\approx 16 \cdot X \cdot Y$	
	VR	$\approx 21 \cdot X \cdot Y$	$\approx 42 \cdot X \cdot Y$	
Design Rule Complexity		$\approx 97 \cdot X \cdot Y$	$\approx 198 \cdot X \cdot Y$	$O\big(X \cdot Y\big)$
Framework Scalablility		$O\big(X \cdot Y \cdot \lvert P \rvert\big) + O\big(X \cdot Y \cdot \lvert D \rvert\big)$		

commodity d_m^n as described in (2.14). Empirically, the numbers of clauses and literals are approximately 96 and 224 times of $X \cdot Y \cdot |T|$, respectively. Meanwhile, the complexity of CFC constraint for pins is different from that of CFC for grids because we refine the formulation using Super Outer Node S_o for all commodities as described in Section 1.4.2. Instead of per commodity d_m^n, we define CFC constraint for S_o per outer pin. As a result, the numbers of clauses and literals for pin-related CFC are approximately 14 and 28 times of $X \cdot Y \cdot |S_o|$, respectively.

By nature of exclusiveness rule, we define VE constraint per each vertex. For grid-related VE, the complexity of each vertex is $O\big(|N|\big)$ due to **AM1** of net indicator as shown in (2.19). As

a result, the complexity of grid-related VE is $O\big(X \cdot Y \cdot |N|\big)$. The observed numbers of clauses and literals for grid-related VE are 30 and 60 times of $X \cdot Y \cdot |N|$. In the case of pin-related VE, our framework only includes constraints for inner pin as explained in (2.18). Therefore, the complexity of pin-related VE is $O\big(|S_i|\big)$ because the number of the adjacent vertices for the inner pin is not changed but limited by the number of routing tracks in a row[2]. Empirically, the numbers of clauses and literals for pin-related VE are 10 and 20 times of $|S_i|$, respectively.

For EA, (2.20) logical implication constraint between the flow indicator $f_m^n(v,u)$ and the edge indicator $e_{v,u}^n$. They must be defined for all commodities and edges, resulting in $O\big(X \cdot Y \cdot |T|\big)$ complexity. For MS, the complexity is $O\big(X \cdot Y \cdot |N|\big)$ because metal segment is the auxiliary variable for net exclusiveness as explained in (2.21).

The overall complexity of the flow formulation is $O\big(X \cdot Y \cdot |P|\big)$ by utilizing the number of pins $|P|$ because $|P|$ is the sum of the number of commodities $|T|$ and the number of nets $|N|$ (i.e., $|P| = |T| + |N|$). Empirically, the numbers of clauses and literals for the flow formulations are approximately 103 and 238 times of $X \cdot Y \cdot |P|$, respectively.

Complexity of Design Rules. For GV, we define geometric variable for all vertices as shown in (2.25). Therefore, the complexity of GV is $O\big(X \cdot Y\big)$. Similarly, the complexity of each design rule is also $O\big(X \cdot Y\big)$ because we define all constraints for design rules per vertex as shown in (2.26-2.34).

Even though we use the "pairwise" encoding for **AM1** of design rules, the overall complexity is maintained as $O\big(X \cdot Y\big)$ because the domain of **AM1** is smaller (e.g., ranging 2-4 in our formulation) than that of flow formulation. In our formulation (including 5 design rules), the observed numbers of clauses and literals are 97 and 198 times of $X \cdot Y$, respectively.

Framework Scalability. Our framework scalability is $O\big(X \cdot Y \cdot |P|\big) + O\big(X \cdot Y \cdot |D|\big)$ (i.e., flow formulation + the number of conditional design rules $\times |V|$) thus many design rules can be representable in practical routabililty analysis.

[2]The number of routing track is 5 in our calculation.

2.4.3 An Example of Routability Analysis Result (ILP & SAT)

Figure 2.11 and Figure 2.12 show examples of the ILP-based routing solution and the SAT-based routing solution, respectively. In Figure 2.11, the metal length is 212 including four extended metal segments (depicted in red-dotted circles) to avoid DRC violations. The SAT-based solution in Figure 2.12 is not optimized (cost = 416), but it only takes 0.02% of ILP's runtime to analyze the design-rule correct routability. Both Figure 2.11 and Figure 2.12 analyses result in "routable".

2.5 Experiments

In this section, we describe the experiments. This section consists of *(i) Experimental Setup*, *(ii) Improvement of Formulation Complexity*, *(iii) Benchmark Information and Switchbox Generation*, *(iv) Experimental Results using Benchmark*, and *(v) Scalability of the Routability Analysis Framework*.

2.5.1 Experimental Setup

We implement our frameworks as a chain of scripts to extract the grid-based switchbox from the LEF/DEF-format layout design (Testcase Generation), generate the ILP/SAT formulation (Routability Analysis), and diagnose using MUS of the unroutable layout (Routability Diagnosis). Our frameworks are validated on a 2.4GHz Intel Xeon E5-2640 with 256GB memory and 12 hyperthreaded CPU cores.

In the analysis framework, we adopt *CPLEX 12.8.0* [6] as our ILP solver, and several state-of-the-art multi-threaded SAT solvers (e.g., *Plingeling bcj* [59], *glucose-syrup 4.0* [60], and *manyglucose-4.1.2* [61]) in the portfolio manner to get the best SAT solving performance in terms of run time. We restrict the maximum number of threads to 12 during experiments for both the ILP-based solver and the SAT-based solver for fair comparisons between the ILP-based routing

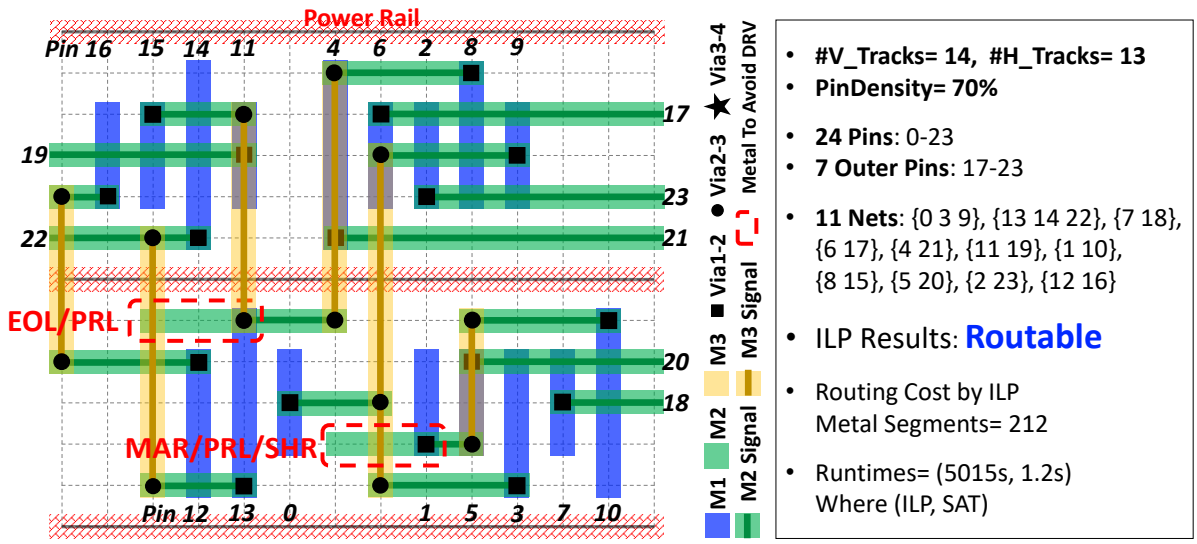


Figure 2.11: The ILP-based optimal routing solution (Cost = 212). Four more metal segments are assigned to avoid DRC violations (red dotted circles).

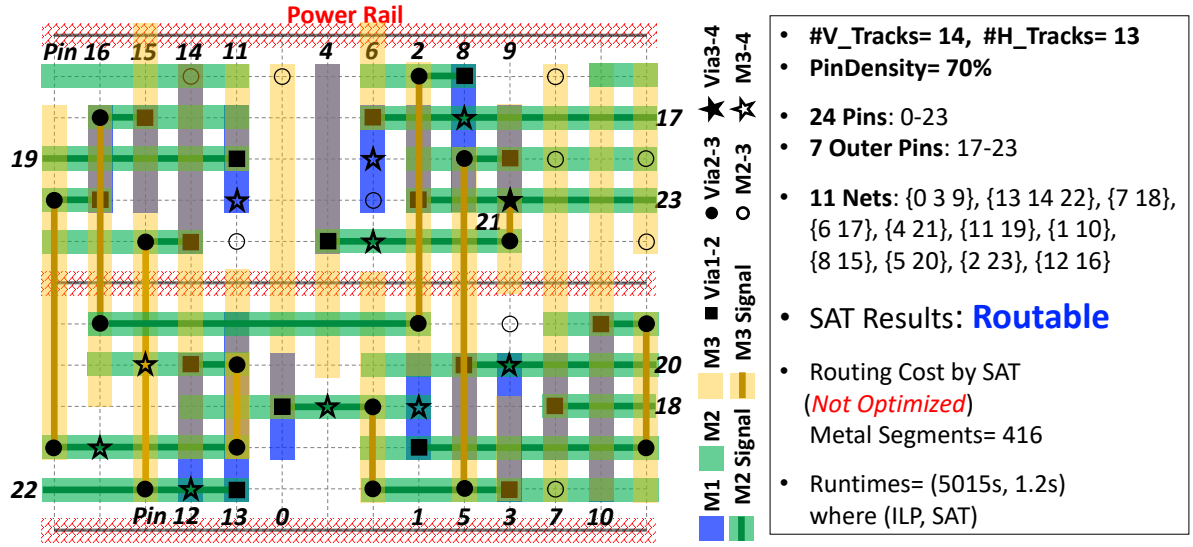


Figure 2.12: The SAT-based routing solution (Cost = 416). The solution is not optimal, but takes only 0.02% of ILP's runtime.

optimization and the SAT-based routability analysis[3].

[3]We limit the maximum execution time to 12 hours and 1 hour for *CPLEX* (ILP solving) and *Plingeling* (SAT solving), respectively.

Table 2.3: Experimental results presenting the ILP-based optimization vs. the SAT-based routability analysis with comparison between [5] and our proposed framework. In the table, red. = reduction ratio, T = runTime (In ILP, *CPLEX* is terminated by the time limit), and T% = runTime% vs. ILP. T% Average = the average T% of the SAT. Design Rules [grids]: MAR/EOL = 2, VR = $\sqrt{2}$, PRL/SHR = 1.

Testcase [5]	Spec.		ILP-based Routing Optimization							SAT-based Routability Analysis									
			#Variables			#Constraints			T(s)	#Variables			#Literals			#Clauses			T(s)	T%
	$\|N\|$	$\|P\|$	[5]	Modified	red.	[5]	Modified	red.		[5]	Modified	red.	[5]	Modified	red.	[5]	Modified	red.		
14_13_60	10.0	21.8	144,721.2	44,314.6	69%	232,174.2	48,167.0	79%	1,076.6	80,315.6	26,345.8	67%	6,807,435.0	302,362.4	96%	3,471,788.6	125,504.0	96%	0.4	0.04
14_13_70	12.0	26.4	195,522.0	53,139.4	73%	312,683.8	56,672.0	82%	3,363.0	107,112.6	34,699.8	68%	12,046,890.0	350,503.6	97%	6,048,136.6	140,793.6	98%	0.4	0.06
14_19_60	14.2	32.2	357,111.8	93,396.4	74%	578,829.2	97,775.8	83%	33,241.4	194,278.6	60,034.0	69%	27,316,733.8	580,761.4	98%	13,743,476.8	230,531.6	98%	1.2	0.01
14_19_70	16.4	36.0	417,015.6	103,795.4	75%	663,016.0	107,065.8	84%	34,677.7	225,426.4	68,852.8	70%	38,228,830.0	668,236.2	98%	19,287,022.2	265,419.0	99%	1.7	0.03
20_13_60	14.6	32.8	368,255.8	92,873.4	75%	595,007.8	97,315.8	84%	35,920.7	199,829.8	59,319.8	70%	29,761,672.6	577,675.2	98%	14,947,286.4	229,584.2	99%	1.0	<0.01
20_13_70	16.4	36.2	415,530.2	101,787.0	76%	662,946.0	106,092.8	84%	>43200	224,503.8	66,902.4	70%	38,550,581.4	646,517.2	98%	19,287,022.2	256,145.2	99%	1.5	<0.01
20_19_60	16.6	44.8	797,507.6	182,478.8	77%	1,293,681.0	186,431.2	86%	35,902.8	427,871.0	123,491.6	71%	95,090,666.4	1,143,092.0	99%	47,596,732.8	449,297.4	99%	9.8	0.03
20_19_70	23.0	52.6	1,037,195.2	213,066.6	79%	1,677,742.8	214,339.2	87%	35,163.2	552,123.0	147,245.0	73%	176,174,452.4	1,354,865.8	99%	87,585,849.8	528,807.4	99%	2.8	0.01
T% Average			(reference) 100.00																	<0.02

2.5.2 Improvement of Formulation Complexity

Table 2.3 presents the experimental results for comparison between [5] and our framework with example layouts of [5] in terms of the formulation complexity and solving performance. Note that each row in Table 2.3 represents five distinct testcases, and shows numbers on average (following [5]). Compared to [5], the number of variables and constraints of ILP formulation are reduced up to 79% and 87% in our framework, respectively. The reduction of ILP formulation comes from undirected multi-commodity flow foundation and supernode simplification scheme. In the case of SAT formulation, the number of variables, literals, and clauses are reduced up to 73%, 99%, and 99% compared to [5], respectively. On top of the reduced ILP formulation, the ILP-to-SAT conversion platform (i.e., SAT encoding and logic minimization) and BCP-based solver can achieve a further reduction in terms of the CNF representation. As a result, our SAT-based routability analysis provides all testcases in Table 2.3 within 10 seconds on average which is $<$0.02% (i.e., more than 5000$\times$ faster) of the ILP runtime on average (ranging from $<$0.01% to 0.06%).

2.5.3 Benchmark Information and Switchbox Generation

We utilize the benchmark of ISPD-2019 contest [62] to validate the scalability of our framework. The benchmark is based on generic 28*nm* cell libraries (i.e., nine-track and off-grid based designs). For a fair comparison between industry detailed routing tool and our framework, all standard cell libraries are mapped to the on-grid routing graph in terms of I/O pin access points and all design rule settings are mapped to our framework's conditional design rules. From the benchmark's DEF, we fully inherit the benchmark's own information related to netlist and placement result of standard cells. With the updated LEF and benchmark's DEF, we perform the detailed routing using *Cadence Innovus v18.10* [4] to obtain an updated DEF including the detailed routing results based on framework-aware LEF (i.e., routed DEF). We generate the grid-based routing graph (i.e., switchbox) with I/O pin layout which is the input of analysis framework based on the routed DEF. When we extract a switchbox from routed DEF, the elements of irrelevant nets are mapped to the obstacles that are the initial assignments for the switchbox.

2.5.4 Experimental Results using Benchmark

Table 2.4 presents the experimental results of SAT-based routability analysis using ISPD-2019 benchmark with various DRV conditions. To utilize benchmark, we first perform the detailed routing functionality[4] of commercial tool to obtain the DRV regions. Based on each DRV region (i.e., switchbox), we perform our routability analysis framework to demonstrate the performance and scalability.

DRC-Clean (Region I). Figure 2.13 shows that our framework successfully generates design rule-correct routing solution (i.e., DRC-Clean) for a DRV-included switchbox (Region I) of industry routing solution as shown in Figure 2.13. By using the same switchbox of commercial routing solution (Figure 2.13(a)), we can get a routable solution within 0.4 seconds and the

[4]We perform iterative trials, which is the 2×20 detailed routing optimizations, to reduce the overall DRC violations.

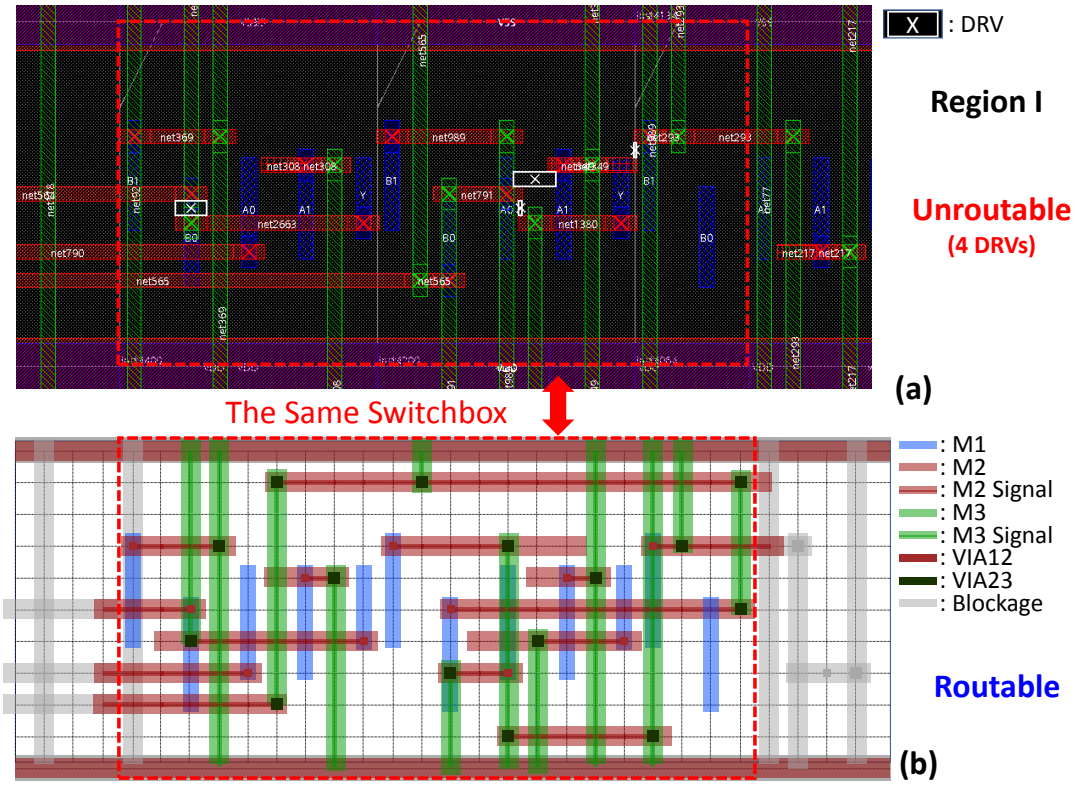


Figure 2.13: An example of switchbox from ISPD 2019 benchmark. (a) DRC violations from the industry routing tool. (b) Optimized routing solution generated by our framework (ILP).

optimized routing solution through ILP as shown in Figure 2.13(b) because our framework applies concurrent modeling to all source-sink connectivities instead of any net-ordering techniques (i.e., sequential approach).

2.5.5 Scalability of the Routability Analysis Framework

Figure 2.14 shows the scalability of the analysis framework. The framework complexity is almost proportional to the product of layout size and the number of pins as explained in Section 2.4.2[5] (Figure 2.14(a)). Our analysis framework has demonstrated its capability handling the complexity with up to $210 \times 211 \times 758$ (i.e., $X \cdot Y \cdot |P|$) of unroutable case ($\approx 0.45B$ clauses). Due to the nature of SAT solving, the runtime T_a depends on the routability type of switchbox as

[5]The complexity of proposed framework can be $O(X \cdot Y \cdot |P|)$ because the framework has 5 design rules (i.e., $|D| = 5$).

Table 2.4: Experimental results presenting the SAT-based routability analysis using benchmark. In the table, Result = Analysis Result (R:Routable, U:Unroutable, UN:Undetermined within 1 Hr). All times in seconds. Design Rules[grids]: MAR/EOL = 2, VR = $\sqrt{5}$, PRL/SHR = 1.

DRV	$G(V,E)$		Spec.		SAT formulation			Analysis	
Region	X	Y	$\|N\|$	$\|P\|$	#Variables	#Literals	#Clauses	Result	T_a
I	22	11	11	28	25,849	305,070	129,541	R	0.4
II	11	11	3	6	2,390	21,904	9,662	U	0.03
	50	51	31	78	711,600	6,223,044	2,499,951		6.5
	100	101	107	306	12,640,796	114,246,609	44,676,894		118.0
	130	131	150	456	31,548,414	287,132,957	111,851,885		299.0
	120	251	203	617	75,905,221	686,564,221	267,125,453		869.8
III	40	21	20	50	181,486	1,622,798	649,429	U	1.3
	50	51	34	96	935,559	8,414,325	3,334,900		13.2
	100	101	73	210	8,377,845	74,828,963	29,433,312		88.1
	150	151	97	383	31,527,855	288,360,468	111,787,511		731.3
	160	151	150	468	43,731,414	395,879,429	154,551,067		327.1
	180	151	163	518	54,701,153	493,161,312	192,196,400		893.0
	200	201	142	599	88,239,306	814,945,216	314,412,425		737.2
	210	211	143	618	99,872,227	923,921,781	356,118,269	UN	
	210	211	229	758	128,603,347	1,164,414,397	452,603,482	U	744.7
	60	21	24	61	314,276	2,816,983	1,122,807	R	295.8
IV	41	21	16	39	104,706	944,948	389,512	U	0.9
	130	21	25	67	525,432	4,481,977	1,821,497		4.2
	240	21	50	139	2,297,158	20,407,862	8,148,453		15.2
	50	21	19	49	165,854	1,414,446	575,577	R (Case1)	46.7
	60	31	24	63	348,492	2,987,555	1,213,592		260.7
	80	31	29	83	597,254	5,141,238	2,082,008		1,100.9
	90	31	30	86	681,806	5,836,976	2,368,057		1,997.3
	140	11	20	48	180,629	1,506,547	618,057	R (Case2)	7.4
	200	11	30	74	373,814	3,122,504	1,277,833		23.1
	200	21	49	126	1,211,678	10,227,342	4,186,623		1,099.8
	60	21	16	40	156,921	1,432,690	587,317	R (Case3)	30.0
	100	21	22	52	358,481	3,094,894	1,252,142		271.1
	150	21	31	80	821,833	7,184,471	2,884,399		1,559.2

shown in Figure 2.14(b). For unroutable case, T_a is linearly increasing as following the complexity of switchbox in the limitation criteria (1hr time limit & tool limit of the SAT solver). On the other hand, T_a of routable cases is rapidly increasing as the complexity is increasing, thus its scalability is limited up to $200 \times 21 \times 126$ case with 1,099.8 seconds of runtime ($\approx 4M$ clauses). In particular, the scalability of routable case depends on the layout architecture (Case-Dependent)

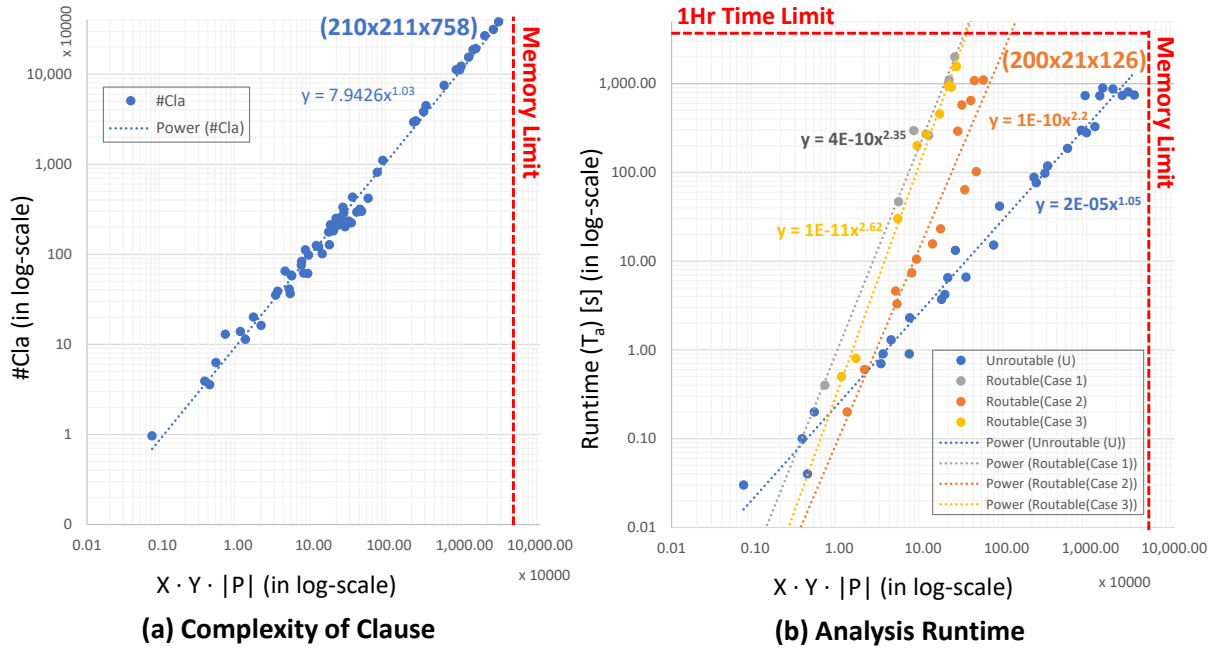


Figure 2.14: Scablablility of our routability analysis framework (in log-scale). (a) Complexity of clause. (b) Runtime (T_a).

as shown in Figure 2.14(b) and Table 2.4.

2.6 Conclusion

In this chapter, we have described a new design methodology for fast turnaround in analyzing the routing feasibility of the given layout (Routability Analysis). Our frameworks efficiently identify the design rule-correct routability by solving the proposed ILP and SAT formulation. We develop a new SAT-friendly ILP-based detailed routing formulation satisfying conditional design rules. During our ILP-to-SAT conversion, we reduce the complexity of the SAT problem by utilizing SAT encoding techniques, logic minimizer, and preprocessing. We demonstrate that our SAT-based routability analysis frameworks produce precise assessment of the design rule-correct routability, within 0.02% of ILP runtime on average. We show that our frameworks handle up to $210 \times 211 \times 758$ case for routability analysis. In next chapter, we describe the details of routability diagnosis framework.

This chapter contains materials from “Grid-based Framework for Routability Analysis and Diagnosis with Conditional Design Rules”, by Dongwon Park, Daeyeal Lee, Ilgweon Kang, Chester Holtz, Sicun Gao, Bill Lin and Chung-Kuan Cheng, which appears in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, February 2020. The dissertation author was the primary investigator and author of these papers.

Chapter 3

Detailed Routing: Routability Diagnosis Framework

3.1 Introduction

In previous chapter, we presented routability analysis framework for fast turnaround in analyzing the routing feasibility through the virtue of SAT's fast reasoning. However, even if we achieve the routability analysis successfully, the diagnosis for the routing failure remains an open question in the detailed routing.

In this chapter, we propose a novel framework which diagnoses explicit reasons for the detailed routing failure while efficiently estimating the routability of layout architectures. To identify conflict geometry and design rules, ROAD converts design constraints of pin layouts to SAT clauses. It then diagnoses design conflicts based on several SAT techniques, such as Minimal Unsatisfiable Subset (MUS), Boolean Constraint Propagation (BCP) and Partial Implication Graph (PIG). Using the MUS technique [23, 63] identifying a minimum unsatisfiable subset of conflicting clauses, we first extract partial layouts that include the causes of conflicts in the unroutable case. BCP [64, 21] is a simplification procedure for clauses using unit clauses. We iteratively perform BCP procedures until either there are no more remained unit clauses or any conflict is found. ROAD then represents variable nodes and clause edges generated by the BCP procedure as a directed acyclic graph (DAG). This subprocedure is conducted by the PIG technique [64], and we exploit the PIG to produce diagnosis results of the conflict layouts.

The main contributions of this paper are listed as follows:

- We propose a framework for routability diagnosis, called *ROAD*, providing comprehensive SAT-based diagnosis information for unroutable layouts, e.g., conflicting geometries and design rules. This information is useful to layout designers for achieving a timely troubleshooting.

- ROAD provides comprehensive information of unroutable layouts, e.g., conflict geometries and design rules. This information is useful to layout designers for fast trouble-shooting and to design-rule managers for fine-grained decisions of design rule priority.

- We demonstrate that our diagnosis procedures are suitable to identify the causes of unroutable pin layouts.

- ROAD utilizes an automated technique [5] which generates pin layout testsets based on *Rent's Rule* [65]. It enables comprehensive exploration of conflicting cases which layout designers encounter in practice. We also demonstrate our ROAD using the practical benchmark utilized in previous chapter.

Our ROAD framework allows designers to diagnose the causes of routing failure cases and provides useful insights with enhancement of understanding for the unroutablity. In this paper, we also present several key findings. We find that layout conflicts happen due to two main factors: (i) pin-shape pattern and (ii) routing resource shortages (i.e., routing congestion). The pin-shape pattern is the main concern of pin accessbility problems. When the resource is insufficient, technology limitation may hinder layout routability, e.g., due to insufficient tracks and metal layers about pin layout. We find the exact geometry and design rules related to the conflict, and thus it can be a guideline how to resolve the conflict.

3.2 Routability Diagnosis Framework

By virtue of the fast reasoning ability of SAT, we suggest the new diagnosis framework to provide useful information for DRVs. This section consists of *(i) Overall flow of Routability Diagnosis Framework*, *(ii) MUS (Minimal Unsatisfiable Subset) Extraction*, *(iii) Decision with Longest-path Search (DLS)*, *(iv) Diagnosis Result Report*, *(v) DRV Classification* and *(vi) Application Scenarios using our frameworks*.

3.2.1 Overall Flow of Routability Diagnosis Framework

Figure 3.1 shows the overview of the routability diagnosis framework based on SAT techniques. When we have DRVs after analysis, we diagnose routing failures of unroutable designs

and reveal causes of the routing failures. As the first step of the diagnosis procedure, we extract MUS (Minimal Unsatisfiable Subset) [23] for finding conflicting regions. Compared to [66], we remove *"Initial Propagation Phase"* step after MUS extraction since the set of *"Initial Propagation Phase"*-related unit clauses **(L)** is completely removed through the built-in preprocessing functionality (i.e., BCP-based SAT solver in analysis framework) before solving SAT. Once the MUS has been recovered, the subsequent decision procedure performs the iterative element selection and propagation until taking the appropriate conflict choice. We select appropriate elements and execute BCP with partial true/false assignments. The propagation phase generates each PIG (Partial Implication Graph) and its combination including variables and clauses (i.e., a DAG is the combination of multiple PIGs). From the DAG (Directed acyclic graph), we obtain a designer-readable interpretation of the conflict geometry and design rules. The diagnosis using clauses is similar to the longest-path-search problem for a DAG [67]. In this paper, the DAG is defined as follows:

Definition 3.2.1. (Diagnosis DAG) The DAG $H(U, D)$ is defined as

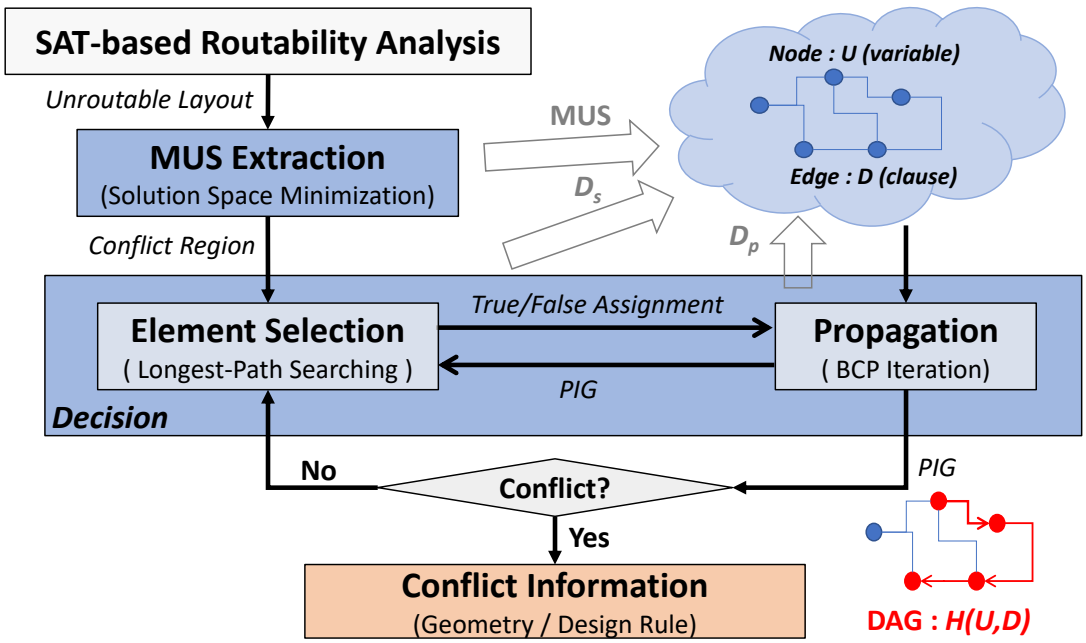


Figure 3.1: Overall flow of our routability diagnosis.

1. D is a set of clauses in a MUS.

2. $U = \{U_s, U_p\} \subset V$ is a set of variables involving the confliction. U_s and U_p are the variables selected by the element selection and propagation phase, respectively.

3. D_s and D_p are a set of unit clauses related with U_s and U_p, respectively.

4. $H(U,D) = PIG(U_s \cup U_p, D_s \cup D_p)$, where $PIG(U,D)$ is generated in each propagation.

3.2.2 MUS Extraction

We first extract a minimal unsatisfiable set (if all its proper subsets are satisfiable) to find the conflict region. Unsatisfiability only involves the MUS of original clauses, thus the MUS significantly reduce the diagnosis time by replacing the diagnosis target with a very small number of clauses (i.e., the domain of MUS). We adopt a state-of-the-art MUS Extractor *MUSer2* [68] as our MUS extractor. Compared to [66], we apply the preprocessing methodology in MUS extraction (i.e., 'SatElite' of *MUSer2*) to accelerate the extraction time [69].

3.2.3 Decision with Longest-path Search (DLS)

In the DLS procedure, we iteratively execute element selection and propagation phase until proper conflict choice. We need to set the combination of the variables to identify the choice of the assignment which enables delivering an appropriate explanation of the conflict region. We adopt a longest-path $\big($i.e., the maximum tree height of $H(U,D)\big)$ search algorithm to choose the most comprehensive conflict choice. The underlying intuition is that the longest routing path usually represents the causes of the routability failure in a comprehensive manner because the longest path carries much richer information related to design rules (i.e., clause D) and geometric information (i.e., variable U) than others. In Figure 3.2, the designer would consider that the short paths depicted in Figure 3.2(a), (b) should be obviously avoided, while the longest path

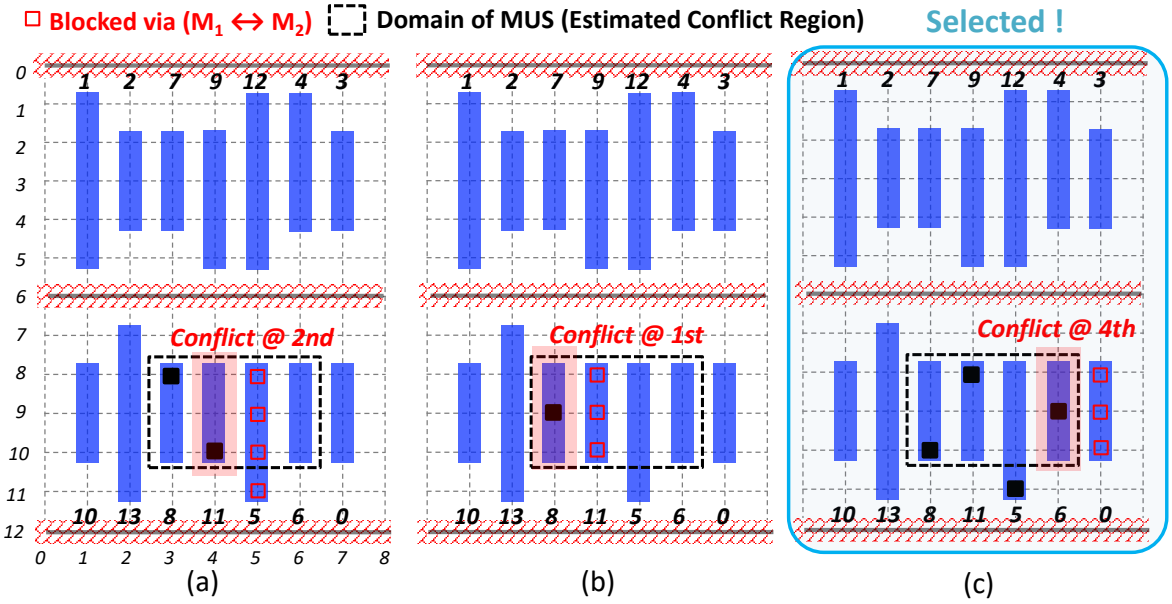


Figure 3.2: An example of DRV explanation based on path length. (a) 2 @ (3,8,1). (b) 1 @ (3,9,1), (c) 4 @ (3,10,1).

in Figure 3.2(c) provides the intrinsic problem caused by the design rules. The detailed DLS algorithm is shown as Algorithm 1 and follows.

Element Selection. Element selection phase (Line 11-17) determines the set of unit clauses (i.e., D_s) with the true assignment for selected edge E. If E is one of the inter-layer via, unit clauses with the false assignment are also generated. The generated D_s is added into the MUS (Line 16).

Figure 3.3 represents elements selection rules for the propagation. The elements selection happens to cover four reasons: *elements for true assignment*, *elements from via-to-via spacing/stacked-via regulation for false assignment*, *void vias in the same pin*, and *void the in-layer elements with direction against true assignment*. Figure 3.3(a) illustrates true assignment scheme for inter-layer via between M_1 and M_2 (i.e., *Via*12). With *Via*12-related elements (1), the elements between the inner pin and *Via*12 are also set as True (2). And the M_2-related elements connected with *Via*12 (i.e., in-layer elements) are set as True as well (3). If E is another inter-layer via (i.e., *Via*23 or *Via*34), the edge to pin (2) cannot be the obvious elements. In the case of

Algorithm 1: DLS

```
1  Input: A MUS of the unroutable layout
2  Output: Conflict information & corresponding DAG H(U,D)
3  while Conflict ∉ H(U,D) do
4      E := GetRemainEdge(MUS);                ▷ Get an unassigned edge e_{v,u}
5      Element_Selection(E);
6      PIG := Propagation(MUS);
7      if PIG.height is maximum among the candidates then
8          Add PIG into H(U,D);
9      end
10 end
11 procedure Element_Selection(E)
12 Add True_Assignment(E) into D_s;
13 if E is inter-layer via then
14     Add False_Assignment(E) into D_s;
15 end
16 Add D_s into MUS;
17 end procedure
18 procedure Propagation(MUS)
19 while MUS has any unit clauses do
20     I := GetUnitClauseList(MUS);
21     BCP(MUS,I) → PIG(U,D);                  ▷ Remove I from MUS
22 end
23 return PIG(U,D)
24 end procedure
```

in-layer edge, we only generate unit clauses related to its edge (3). By the via-to-via spacing case, vias within VR rule from the $Via12$ are blocked. And the stacked-via case blocks via positions which are on top of another via as depicted in Figure 3.3(b). In Figure 3.3(c), only one via is allowable in a pin. In-layer elements, which direction is against true assignment, are blocked from Figure 3.3(d) rule when the net taking $Via12$ is 2-pin net.

Propagation. In the propagation phase (Line 18-24), we execute BCP using the new MUS modified by element selection until there are no more unit clauses in MUS. Figure 3.4 shows an example of the propagation with the true assignment and its part of the PIG. Three via positions $\big($i.e., $m_{(4,9,1)(4,9,2)}$, $m_{(4,10,1)(4,10,2)}$, and $m_{(3,10,2)(3,10,3)}\big)$ are blocked by the VR rule through the

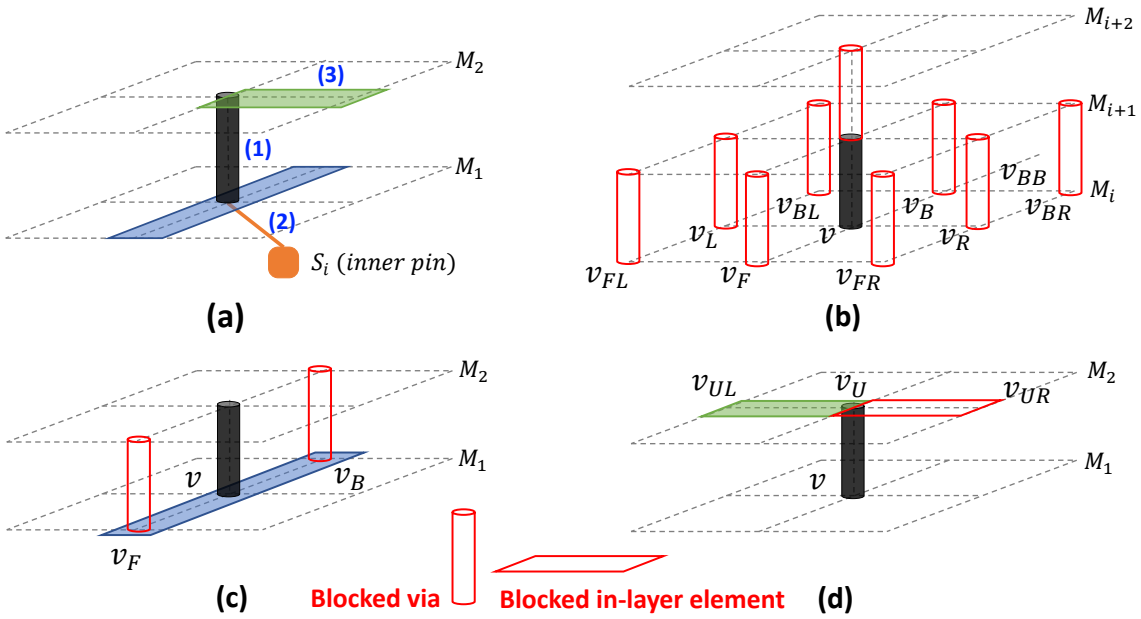


Figure 3.3: Elements selection for the propagation. (a) Elements for true assignment. (b) Elements from via-to-via spacing/stacked-via regulation. (c) Vias in the same pin. (d) In-layer Elements with the direction against true assignment for false assignment.

propagation. Figure 3.5 illustrates an example of propagation with the false assignment and its part of the PIG. The several $Via12$ and in-layer elements are blocked by the false assignment as depicted in Figure 3.5(a). As a result, the propagation selects an E for the next element selection phase among the $Via12$ candidates in $pin11$ since all other candidates are already blocked by the propagation.

3.2.4 Diagnosis Result Report

Our diagnosis framework performs the iterative DLS procedures until it exposes a conflict, resulting in the precise information for the conflict in terms of geometry and conflicted conditional design rules. Figure 3.6 shows the result of diagnosis from our framework. The cause of the unroutability is the conflict between VR and CFC rules due to $f_0^7(pin0)(7,10,1)$ variable. From VR rule, all possible via locations in $pin0$ vertices are blocked $\big($i.e., $f_0^7(pin0)(7,10,1)$ is set as False$\big)$. However, at least one vertex must have a via for $pin0$ connection to meet CFC

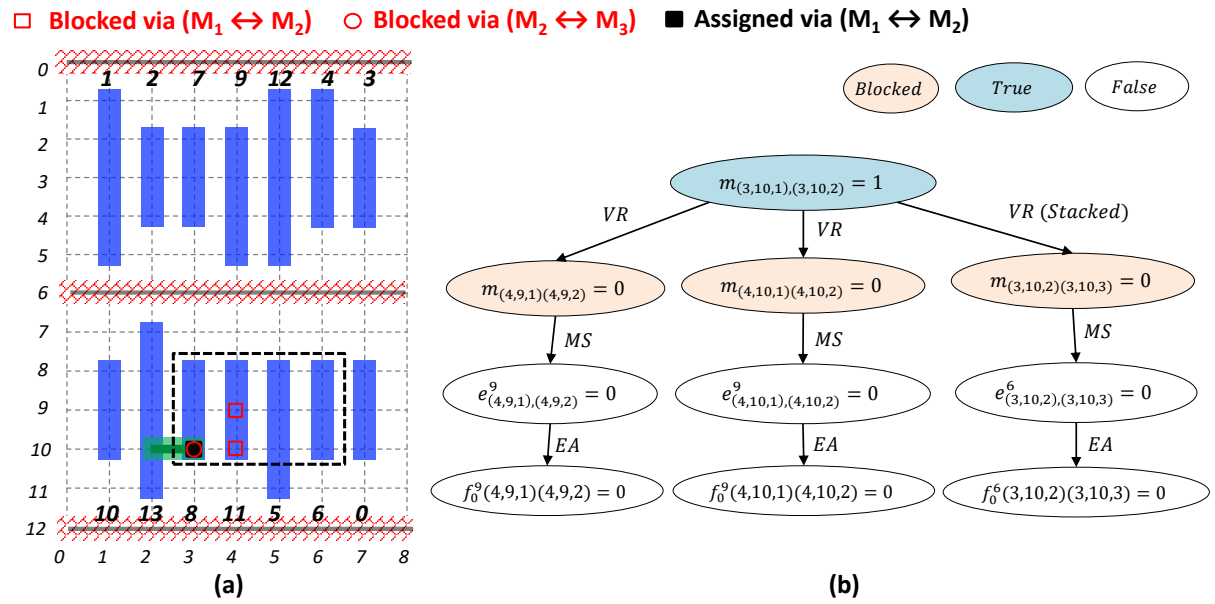


Figure 3.4: An example of propagation with true assignment. (a) Propagation result. (b) Some part of PIG.

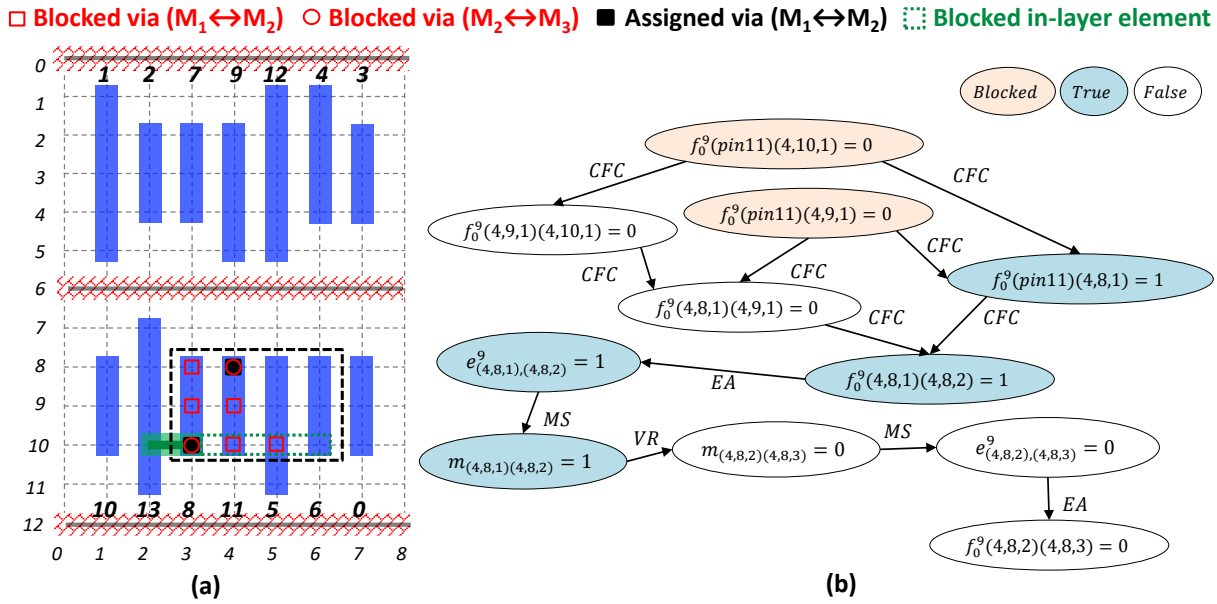


Figure 3.5: An example of propagation with false assignment. (a) Propagation result. (b) Some part of PIG for via-to-via spacing void.

rule (i.e., $f_0^7(pin0)(7,10,1)$ is set as True). At this conflict, we provide a diagnosis report including geometric $\big($i.e., the edge between $pin0$ and $(7,10,1)\big)$ and design rule (i.e., CFC and VR) information about the DRV.

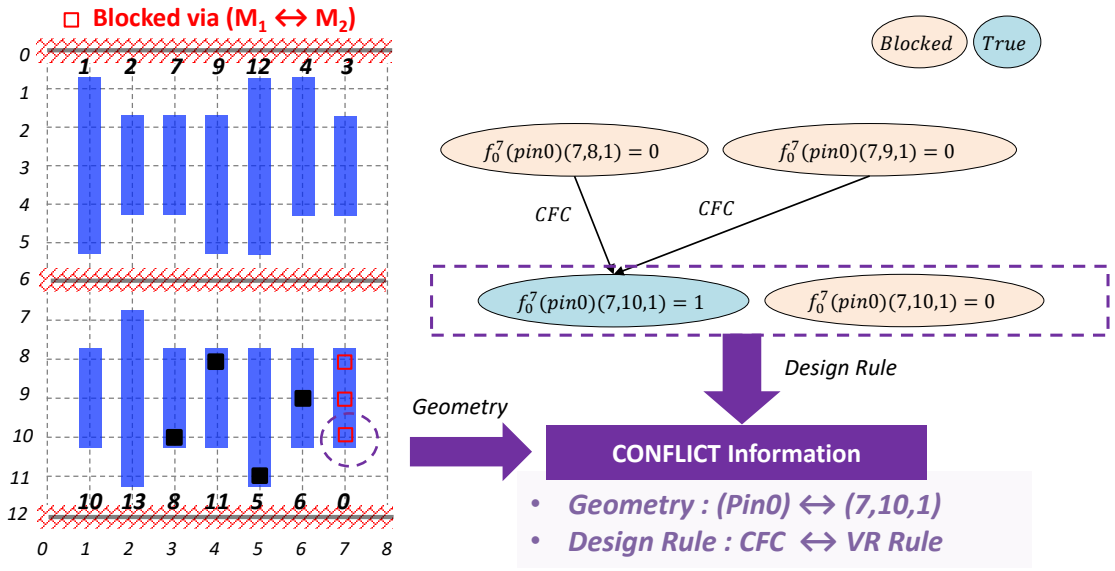


Figure 3.6: An example of diagnosis result.

3.2.5 DRV Classification

Based on the diagnosis result, we can classify the causes of DRV into two categories [66]: *Conflict Pin-shape (CP)* and *Routing Congestion (RC)*. The CP type is one of the main issues in the pin-accessibility problem while the RC type comes from the lack of routing resources (e.g., the number of tracks and metal layers).

Conflict Pin-shape (CP). The CP-related DRV is the problem related to the pin-accessibility of the standard cell. There are two kinds of CP types that are "intrinsic CP" and "obstacle CP". The intrinsic CP type has an apparent CP pattern, while it is definitely infeasible without any further investigation, such as '3-3-n-3-3' pattern as shown in Figure 3.7(a). The obstacle CP type has no intrinsic CP pattern but some pin access positions are blocked by an obstacle (e.g., pre-routed elements) of switchbox. For example, Figure 3.7(b) has a simple '3-3-3' CP pattern induced by the obstacle.

Routing Congestion (RC). The RC-related unroutable layout has no violations in locating vias on the pins in M_1. The real cause of the RC type is related to the lack of routing resources, indicating that the layout requires better congestion handling, placement migration, etc. For

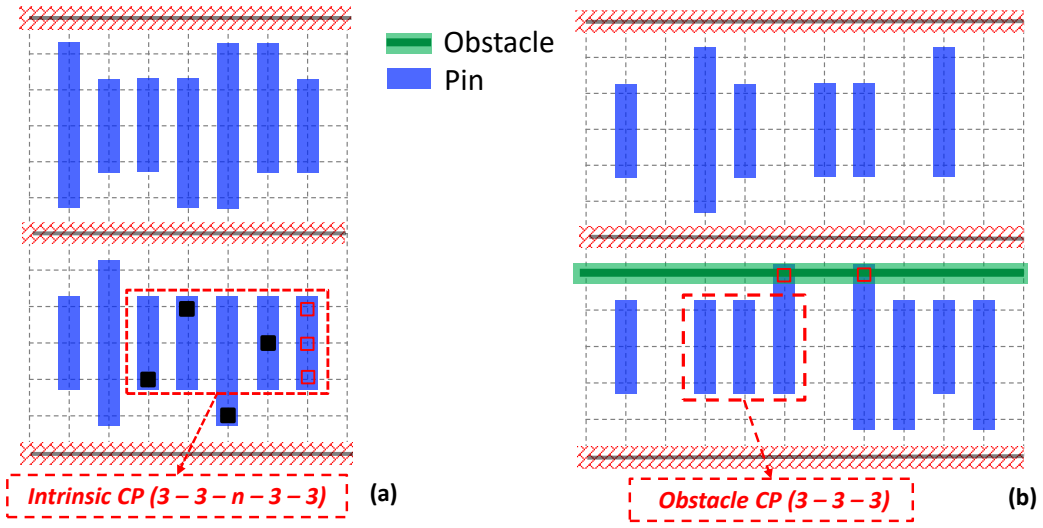


Figure 3.7: Examples of Conflict Pin-shape (CP) type (VR = $\sqrt{2}$). (a) Intrinsic CP. (b) Obstacle CP.

example, the layout in Figure 3.8 has no violations in locating any *Via*12 (Figure 3.8(a)). Net 5 of the layout is one of the in-switchbox connections, and its pins (i.e., pin 1 and 4) should be connected through the inside routing tracks. However, all tracks, of which column coordinate is between 7 and 9 on M_2 layer, are either already occupied by other connections or blocked by design rules after several propagations as shown in Figure 3.8(b), resulting in no ways to connect pin 1 to pin 4.

3.2.6 Application Scenarios using our frameworks

Both routability analysis framework and routability diagnosis framework can be utilized by an ECO enhancement methodology as depicted in Figure 2.1. Due to the limitation of our framework's scalability, we only cover the routability analysis and diagnosis for a switchbox and thus the routability analysis/diagnosis for between switchboxes and for the whole design layout in detailed routing remains open questions.

Figure 3.9 presents the detailed practical scenarios of ECO enhancements from our frameworks. Based on the our diagnosis report for each DRV, we can classify each DRVs into 4 kinds of DRV region as follows. For the Region I, we can resolve DRVs, come from the heuristic

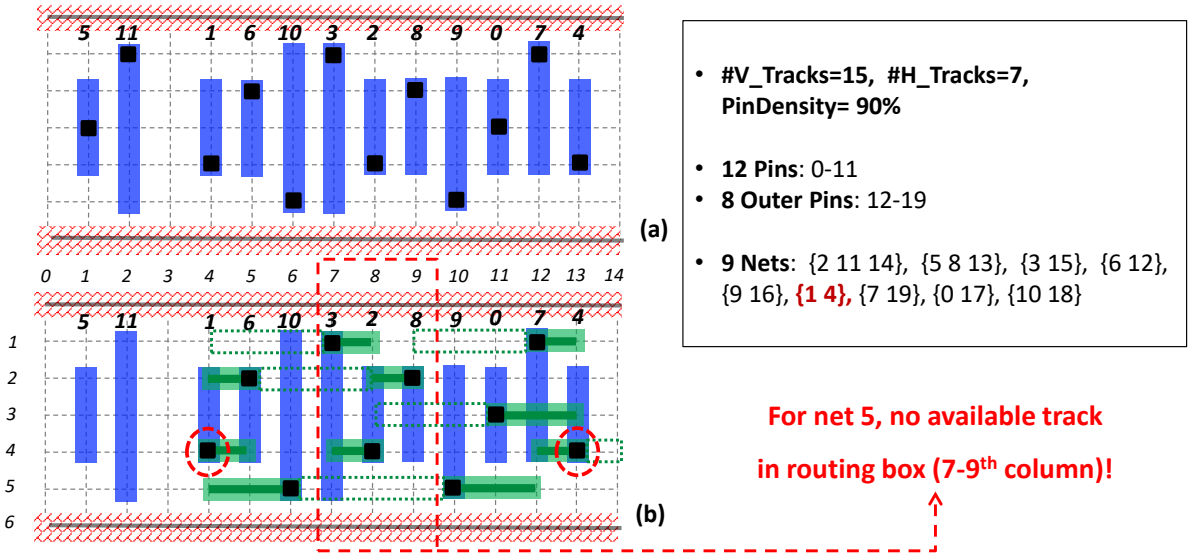


Figure 3.8: An example of Routing Congestion (RC). (a) No violation in locating vias between M_1 and M_2. (c) Lack of routing tracks for net 5.

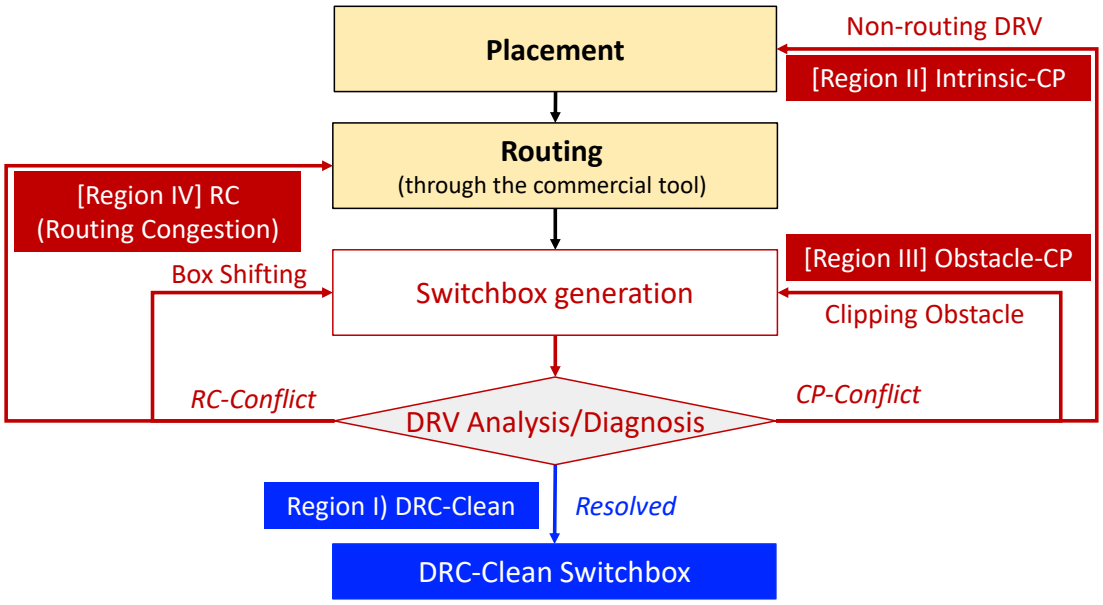


Figure 3.9: Practical ECO scenarios using our frameworks. (Region I) DRC-Clean switchbox. (Region II) Non-routing DRV. (Region III) Switchbox regeneration. (Region IV) Routing-related DRV.

limitation of the commercial tool, since our routability analysis is the exact solving. For the Region II, we provide the information of CP-related DRV to the placement stage. For the Region III, we can regenerate the switchbox by clipping the obstacle (i.e.,the prerouted elements can

be converted into the target nets in the new switchbox). For the Region IV, we can shift the switchbox clipping location for resolving the RC-related DRV. If the RC-related DRV cannot be resolved by several iterations of switchbox shifting, we provide the RC information to the global routing stage for changing the global routing strategy.

3.3 Experiments

In this section, we describe the experiments. This section consists of *(i) Experimental Setup*, *(ii) Experimental Results using Benchmark*, and *(iii) Scalability of the Routability Diagnosis Framework*.

3.3.1 Experimental Setup

By nature of the heuristic approach of MUS extraction in the diagnosis framework, *MUSer2* [68] provides several SAT solvers with preprocessing (e.g., *glucoses*, *minisats*, *minisat-ghs*, and *picosat*) for the foundation of MUS extraction. We pick the best result of extraction in terms of the MUS domain's size and the extraction time from the combinations of each SAT solver and *MUSer2* (i.e., portfolio approach).

3.3.2 Experimental Results using Benchmark

Table 3.1 presents the experimental results of SAT-based routability analysis and diagnosis using ISPD-2019 benchmark with various DRV conditions. To utilize benchmark, we first perform the detailed routing functionality[1] of commercial tool to obtain the DRV regions. Based on each DRV region (i.e., switchbox), we perform our routability analysis and diagnosis framework to demonstrate the performance and scalability.

[1]We perform iterative trials, which is the 2×20 detailed routing optimizations, to reduce the overall DRC violations.

Table 3.1: Experimental results presenting the SAT-based routability analysis/diagnosis using benchmark. In the table, RES. = Analysis Result (R:Routable, U:Unroutable, UN:Undetermined within 1 Hr), T_a = analysisTime, T_m = MUS extractionTime, T_d = diagnosisTime using MUS. All times in seconds. Design Rules[grids]: MAR/EOL = 2, VR = $\sqrt{5}$, PRL/SHR = 1.

DRV	$G(V,E)$		Spec.		SAT formulation			Analysis		Diagnosis			
Region	X	Y	$\|N\|$	$\|P\|$	#Variables	#Literals	#Clauses	RES.	T_a	#MUS	T_m	T_d	Causes
I	22	11	11	28	25,849	305,070	129,541	R	0.4	DRC-Clean			
II	11	11	3	6	2,390	21,904	9,662	U	0.03	43	0.02	0.3	Intrinsic CP
	50	51	31	78	711,600	6,223,044	2,499,951		6.5	67	14.7	0.4	
	100	101	107	306	12,640,796	114,246,609	44,676,894		118.0	82	361.8	0.4	
	130	131	150	456	31,548,414	287,132,957	111,851,885		299.0	79	1,487.5	0.4	
	120	251	203	617	75,905,221	686,564,221	267,125,453		869.8	Tool Limit			
III	40	21	20	50	181,486	1,622,798	649,429	U	1.3	87	5.7	0.4	Obstacle CP
	50	51	34	96	935,559	8,414,325	3,334,900		13.2	87	26.8	0.7	
	100	101	73	210	8,377,845	74,828,963	29,433,312		88.1	107	315.1	0.7	
	150	151	97	383	31,527,855	288,360,468	111,787,511		731.3	109	1,239.9	0.3	
	160	151	150	468	43,731,414	395,879,429	154,551,067		327.1	101	2,143.6	0.6	
	180	151	163	518	54,701,153	493,161,312	192,196,400		893.0	123	2,674.8	0.9	
	200	201	142	599	88,239,306	814,945,216	314,412,425		737.2	Tool Limit			
	210	211	143	618	99,872,227	923,921,781	356,118,269	UN		Tool Limit			
	210	211	229	758	128,603,347	1,164,414,397	452,603,482	U	744.7				
	60	21	24	61	314,276	2,816,983	1,122,807	R	295.8	DRC-Clean			
IV	41	21	16	39	104,706	944,948	389,512	U	0.9	896	3.7	2.0	RC
	130	21	25	67	525,432	4,481,977	1,821,497		4.2	979	17.0	3.5	
	240	21	50	139	2,297,158	20,407,862	8,148,453		15.2	1,431	90.8	7.1	
	50	21	19	49	165,854	1,414,446	575,577	R	46.7	Case1 (DRC-Clean)			
	60	31	24	63	348,492	2,987,555	1,213,592		260.7				
	80	31	29	83	597,254	5,141,238	2,082,008		1,100.9				
	90	31	30	86	681,806	5,836,976	2,368,057		1,997.3				
	140	11	20	48	180,629	1,506,547	618,057	R	7.4	Case2 (DRC-Clean)			
	200	11	30	74	373,814	3,122,504	1,277,833		23.1				
	200	21	49	126	1,211,678	10,227,342	4,186,623		1,099.8				
	60	21	16	40	156,921	1,432,690	587,317	R	30.0	Case3 (DRC-Clean)			
	100	21	22	52	358,481	3,094,894	1,252,142		271.1				
	150	21	31	80	821,833	7,184,471	2,884,399		1,559.2				

CP-related DRV (Regions II and III). Figure 3.10 shows that DRV regions (Regions II and III) containing CP-related DRVs, i.e., the intrinsic CP (Figure 3.10(a)) and the obstacle CP (Figure 3.10(b)) (from the commercial router). As shown in Table 3.1, our analysis framework successfully provides the routability analysis within 6.5 and 13.2 seconds for the switchbox with 50 vertical- and 51 horizontal-track (i.e., 50×51) of Regions II and III, respectively. About the cause of the DRV Regions II and III, our diagnosis framework successfully finds the pin-access

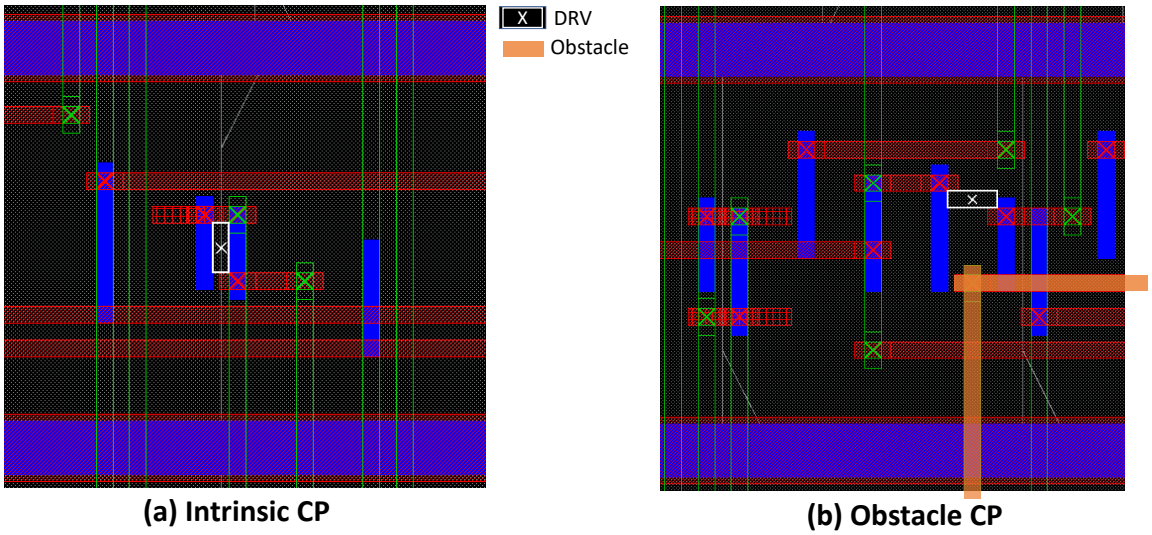


Figure 3.10: Examples of CP-related DRV in the benchmark (VR = $\sqrt{5}$). (a) Intrinsic CP (Region II). (b) Obstacle CP (Region III).

failure at the propagation phase of *Via*12 element within 15.1 and 27.5 seconds (i.e., $T_m + T_d$), respectively. In the case of the obstacle CP-related failure, the routability of the DRV region can be changed from "unroutable" to "routable" if the obstacle is removed (e.g., the $60 \cdot 21$ case of Region III in Table 3.1).

RC-related DRV (Region IV). The 41×21 case of Region IV has only RC-related DRVs because all DRVs have been removed (i.e., routable) when the sizes of switchboxes are enlarged to 50×21. Our diagnosis framework successfully identifies the cause of the DRV as the lack of routing track due to an in-switchbox connection. This information helps layout designers to stop further time-consuming design efforts for routability analysis. As suggested in Section 3.2.6, RC-related DRV can be resolved by shifting the switchbox as shown Table 3.1 (DRV-Clean Case1).

3.3.3 Scalability of the Routability Diagnosis Framework

The scalability of the diagnosis framework is mainly determined by MUS extraction performance as shown in Table 3.1 and Figure 3.11. Our diagnosis framework has demonstrated

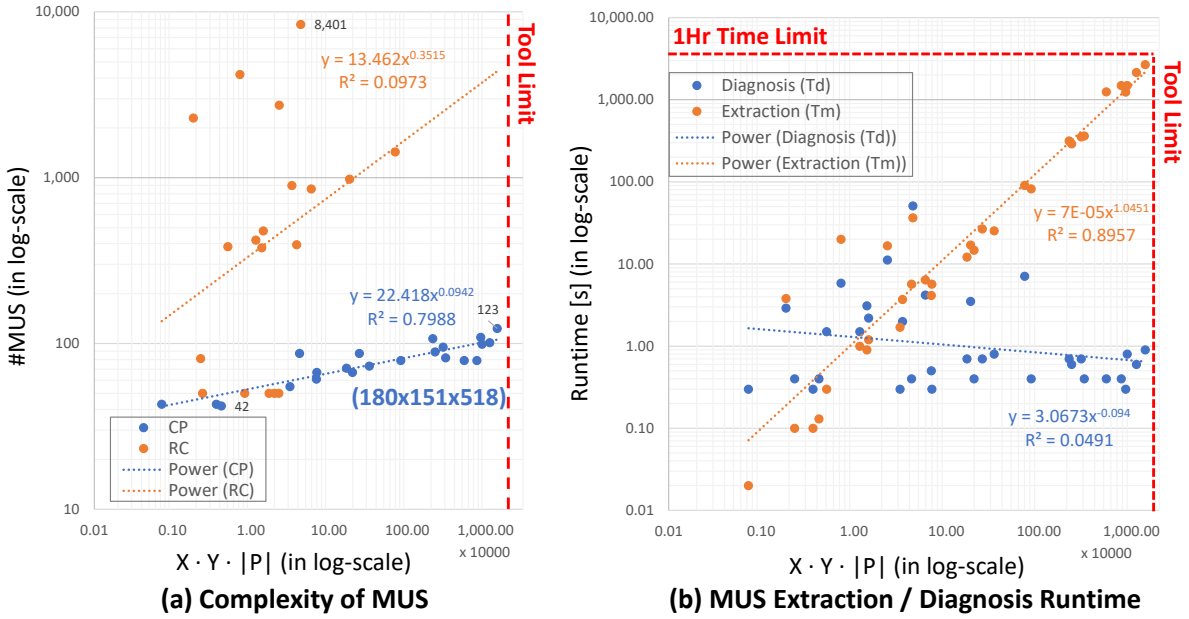


Figure 3.11: Scablablility of our routability diagnosis framework (in log-scale). (a) Complexity of MUS. (b) Runtime (T_m, T_d).

its capability handling the complexity with up to $180 \times 151 \times 518$ case ($\approx 0.2B$ clauses). The complexity of MUS highly depends on DRV type as shown in Figure 3.11(a). For the CP-related DRV, the number of MUSes is not proportional (ranging from 42 to 123) to the complexity of switchbox because the MUS only includes clauses related to the region of CP. For the RC-related DRV case, the number of MUSes is larger than the number of CP-related DRV because the RC-related DRV requires more information (e.g., clauses for geometry and design rules) to explain the cause. But there is no explicit trend related to the complexity of switchbox (P-value: $0.194 > 0.05$). As the complexity of switchbox is increasing, the MUS extraction time T_m is also linearly increasing (P-value: $2.8 \times 10^{-17} \ll 0.05$) in the limitation criteria as shown in Figure 3.11(b). Meanwhile, the diagnosis time T_d has no trends related to the complexity of switchbox (P-value: $0.207 > 0.05$) as shown in Figure 3.11(b).

3.4 Conclusion

We have described a new design methodology for fast turnaround in analyzing the routing feasibility of the given layout (Analysis) and a novel methodology to provide comprehensive diagnosis information of DRVs (Diagnosis). Our frameworks efficiently identify the design rule-correct routability by solving the proposed ILP and SAT formulation. We develop a new SAT-friendly ILP-based detailed routing formulation satisfying conditional design rules. During our ILP-to-SAT conversion, we reduce the complexity of the SAT problem by utilizing SAT encoding techniques, logic minimizer, and preprocessing. We demonstrate that our SAT-based routability analysis and diagnosis frameworks produce precise assessment/diagnosis of the design rule-correct routability, within 0.02% of ILP runtime on average. We show that our frameworks handle up to $210 \times 211 \times 758$ and $180 \times 151 \times 518$ case for routability analysis and diagnosis, respectively.

This chapter contains materials from following publications: "Grid-based Framework for Routability Analysis and Diagnosis with Conditional Design Rules", by Dongwon Park, Daeyeal Lee, Ilgweon Kang, Chester Holtz, Sicun Gao, Bill Lin and Chung-Kuan Cheng, which appears in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, February 2020. "ROAD: Routability analysis and diagnosis framework based on SAT technique", by Dongwon Park, Ilgweon Kang, Yeseong Kim, Sicun Gao, Bill Lin, and Chung-Kuan Cheng, which appears in International Symposium on Physical Design, April 2019. The dissertation author was the primary investigator and author of these papers.

Chapter 4

Standard Cell Synthesis: Simultaneous Transistor Placement and In-cell Routing

4.1 Introduction

In Chapter 2 and Chapte 3, we discussed routability assessment methodologies in DR, which stage deals with routing problem between cells (i.e., inter-cell routing). In this chapter, we propose the methodology for transistor placement and in-cell routing problem in standard cell synthesis.

Standard cell synthesis has become a critically challenging problem because technology nodes of the integration process are continuously shrinking below 7*nm*. In particular, the gap between device and metal pitches becomes larger in sub-7*nm* technology nodes, so the number of available routing tracks per each row (i.e., cell height) is much smaller [2]. Consequently, sets of conditional design rules are newly introduced and/or modified, ensuring manufacturable layout designs on sophisticated multiple-patterning technologies such as LELE (litho-etch-litho-etch), SADP (self-aligned double patterning), and SAQP (self-aligned quadruple patterning) [70]. As a result, layout-design complexity has been significantly increased. To improve PPAC (performance, power, area, and cost) trade-off, the automation of standard cell-layout design recently obtain essential roles for achieving seamless technology transition and design-based equivalent scaling through manufacturability-aware standard cell layout design optimization [2, 71, 72, 73]. However, designing an optimal standard cell layout is nontrivial and extremely laborious since it requires to explore enormously large search space highly-combined with complicated constraints of transistor-level placement and in-cell routing. Due to the difficulties discussed above, most of the previous works focus on divide-and-conquer-style sub-problems and/or heuristic approaches, limiting the solution space and optimality.

4.1.1 Related Works

Standard Cell Synthesis. For transistor-level placement problem, many approaches have been proposed to reduce the search space by adopting heuristic approaches such as “Eulerian

trail" [74][75], "Branch and Bound" [76], "Transistor connection pruning" [77], etc. For in-cell routing problem, several approaches based on traditional "Maze Routing algorithm" [78][79] are suggested but inapplicable to modern multiple-patterning technologies. An SADP-aware routing solution has been presented [80], and several pin-accessibility optimization techniques have attracted considerable attention to improve the pin-accessibility of standard cells in sub-7*nm* technology [80, 37, 42, 13]. However, these approaches relying on solving sub-problems are hard to reach the optimal solution of standard cell layout because of the intractable search space partitioning and the intrinsic limitation of heuristic methodology. For the automation of standard cell layout design procedure, a few works [81][82] co-optimizing both transistor-level placement and in-cell routing together are published. However, these approaches are not suitable for the multiple-patterning technologies in sub-7*nm*. Recently, a sub-7*nm* applicable standard cell synthesis automation frameworks have been proposed [8, 83, 84]. However, these works include sequential/heuristic approaches in place-and-route phase.

Satisfiability Modulo Theories (SMT). Compared to SAT (Boolean satisfiability), SMT is a more expressive language containing non-Boolean variables (e.g., integer, bit-vector, etc.) and predicate symbols as described in [24]. Recently, several SMT (Satisfiability modulo theories) solvers including the optimization methodology (i.e., OMT) are released [31, 32]. By virtue of SAT's fast reasoning ability, SMT-based methodology empowers us to represent the given standard cell layout design problem with much richer modeling language.

4.1.2 Our Contributions

In this chapter, we propose a novel SMT-based framework that **S**imultaneously optimizes **P**lace-**&**-**R**oute (*SP&R*) of standard cell layout in the highlight of practical design features and the improved scalability, resulting in the generation of a whole set of a standard cell library. Our contributions are as follows:

- We propose an automated standard cell synthesis framework, *SP&R*, which simultaneously solves place-and-route (P&R) optimization problem. We devise an innovative dynamic pin allocation (DPA) scheme to integrate placement and routing steps into an optimization procedure.
- *SP&R* utilizes an SMT (Satisfiability Modulo Theories) solver, capable of SAT-based fast reasoning with an OMT (Optimization Modulo Theories) feature of multi-objective optimization.
- *SP&R* covers a wide variety of conditional design rules for DFM (design for manufacturing), e.g., multiple-patterning techniques, producing pin-accessibility-aware cell layouts.
- *SP&R* provides practical cell-design features to further optimize cell sizes and secure the stable operation of timing-critical sequential cells. For example, the use of a single diffusion break with a crossover and a crosstalk mitigation of timing critical nets.

The remaining sections are organized as follows. Section 4.2 introduces the preliminaries of framework. Section 4.3 describes constraint formulation for the simultaneous place-and-route multi-objective problem. Section 5.2 presents the scalability improvement techniques. Section 4.4 discusses our experimental setup/results. Section 4.5 concludes the paper.

4.2 Framework Preliminary

This section introduces an overview of the proposed *SP&R* framework, SMT (satisfiability modulo theories), multi-objective optimization, and target cell architecture.

4.2.1 Overview of SP&R Framework

We formulate a conventional (sequential) standard cell layout process as a constraint satisfaction problem (CSP) with variables and constraints to integrate place-and-route steps into

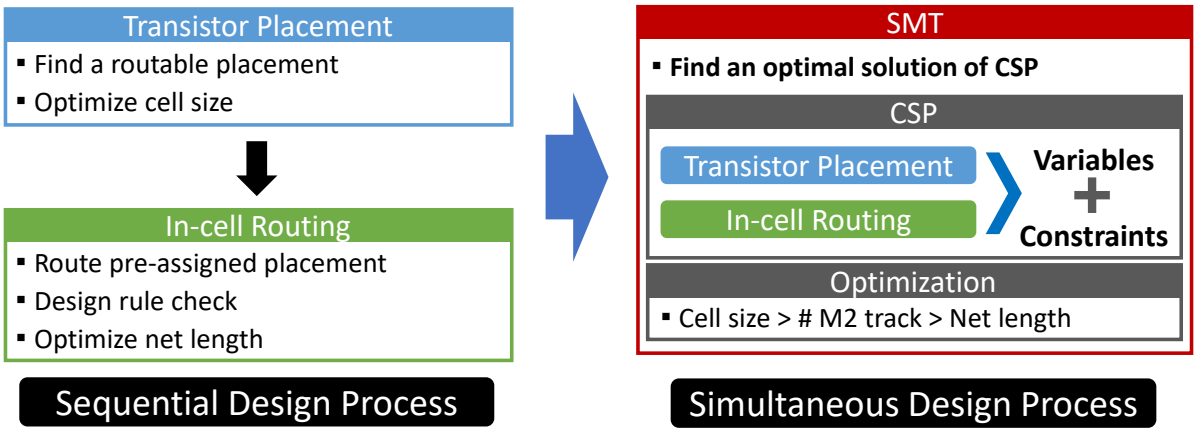


Figure 4.1: Sequential vs. our proposed simultaneous cell design processes.

a multi-objective optimization problem as shown in Figure 4.1. We adopt the state-of-the-art *lazy*-approach SMT solver *Z3* [31, 85] to solve the given optimization problem. Figure 4.2 illustrates an overview of our *SP&R* framework. Given netlist information and a cell architecture, our framework simultaneously obtains an optimal solution that strictly satisfies the constraints of transistor placement, in-cell routing, and conditional design rules. The individual placement and routing problems are combined by our novel dynamic formulation for conditional pin allocation (i.e., DPA).

4.2.2 SMT (Satisfiability Modulo Theories)

On top of efficient problem-solving ability of SAT, SMT provides the feature of OMT (Optimization Modulo Theories) [31, 32] to obtain an optimal solution. Furthermore, SMT formulas support a much richer modeling language (e.g., "if-then-else" for the "Either-Or" constraint, built-in Boolean cardinality functions such as "at-most k" and "at-least k", etc.) than is possible with SAT or ILP (Integer Linear Program) formulas. These key features of SMT efficiently accomplish exhaustive searching for the optimal solution with the concise expressions of constraints. Sections 4.3 and 5.2 respectively describe our methodology to develop SMT formulation of constraints for *SP&R* and our technique to improve *SP&R*'s scalability.

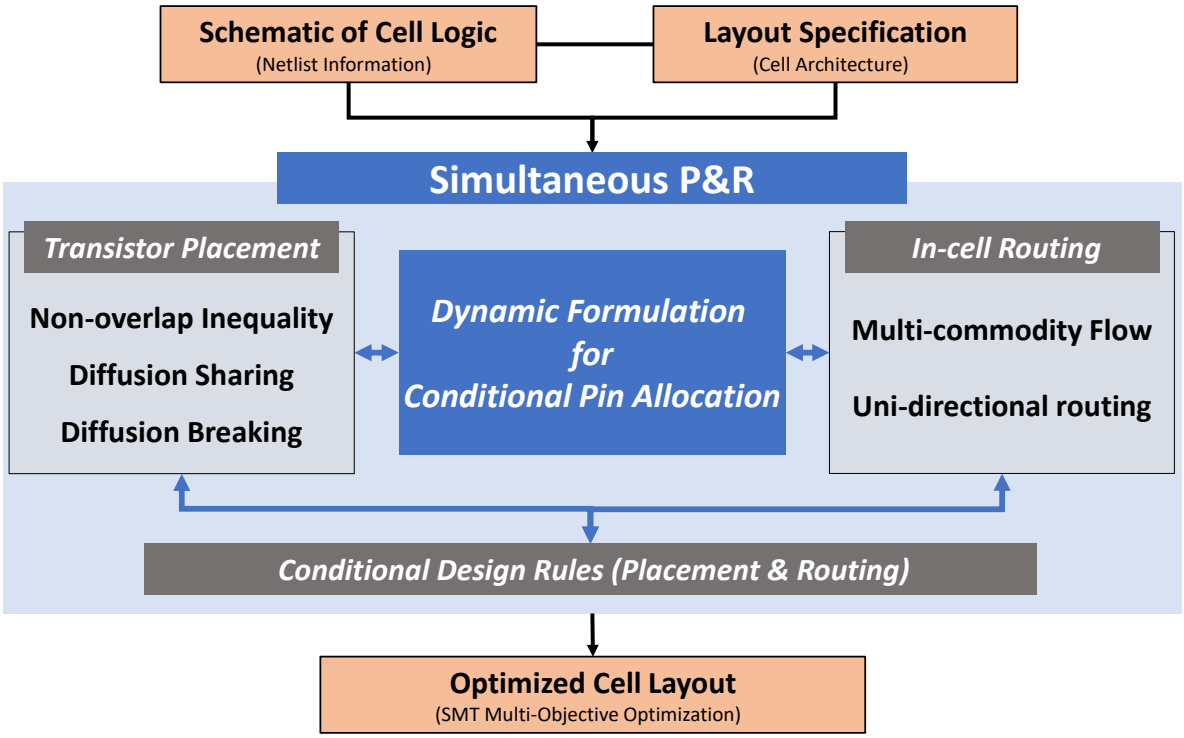


Figure 4.2: The proposed **S**imultaneous **P&R** framework.

4.2.3 Multi-Objective Optimization

SP&R has multiple objectives associated with placement and routing problems for standard cell layout design. The cell size is defined as the maximum occupation of vertical tracks by FETs (field-effect transistors) as shown in Expression (4.1). The number of $M2$ (i.e., top-most metal layer) tracks is defined as the number of occupied $M2$ routing tracks in a generated cell (Expression (4.2)). The total metal length (ML) is the weighted sum of routed metal segments as shown in Expression (4.3). In practice, the cell size has the highest priority because it has a direct impact on the footprint area of a whole chip. Then the number of $M2$ tracks has been used as a more important metric than Total ML; (i) to enhance the cell's PPAC trade-off by suppressing the usage of higher metal layers and (ii) to maximize the routability in detailed routing phase by reserving upper routing resources. Therefore, *SP&R* simultaneously optimizes these multiple objectives in the light of "lexicographic" order with an optimization feature of OMT [86, 31].

The objectives can be ranked in the order of emphasized, as described in Expression (4.4).

$$\textbf{Placement (Cell Size) :}\ \max\left\{x_t + w_t \ \middle|\ t \in T\right\} \tag{4.1}$$

$$\textbf{Routability (\#}M2\textbf{ Track) :}\ \sum_{k=1}^{l} \bigvee_{e_{v,u} \in E_k} m_{v,u} \tag{4.2}$$

l=#Horizontal Tracks

E_k=Set of $M2$ Layer Edges in k_{th} Track

$$\textbf{Routing (Total ML) :}\ \sum_{e_{v,u} \in E} \left(w_{v,u} \times m_{v,u}\right) \tag{4.3}$$

LexMin: (a) Cell Size, (b) #$M2$ Track, (c) Total ML (4.4)

Objectives priority in lexicographic order. *SP&R* co-optimizes multiple objectives at once by using the lexicographic method [31][86]. The lexicographic method consists of solving a sequence of single-objective optimization problems under the constraining condition that optimizes higher-priority objectives. This results in gradual reductions of the search space by virtue of the implicitly added constraints. Therefore, partitioning an objective function with a proper priority helps to improve the scalability. The total metal length objective (described in Expression (4.3)) is defined as the weighted sum of metal segments (i.e., VIA, metal). The weight of the VIA is set higher than the metal to minimize the use of upper-layer metals as well as the use of more resistive VIA elements. Thus, this weight can be used to separate and optimize the

total metal length objective with the priority as shown in Expression (4.5).

$$\textbf{LexMin: } (a)\ \#CA,\ (b)\ \#VIA01,\ (c)\ \#VIA12,\ (d)\ \#M0,\ (e)\ \#M1,\ (f)\ \#M2 \tag{4.5}$$

4.2.4 Cell Architecture

In this work, our framework considers 7*nm* standard cell architecture (e.g., the layer/track information) of [8, 3] as depicted in Figure 4.3. Inspired by [5, 66, 87], we adopt supernodes to cover the multiple candidates for each pin, either the pin of FET (i.e., internal pin) or the I/O pin of a standard cell (i.e., external pin).

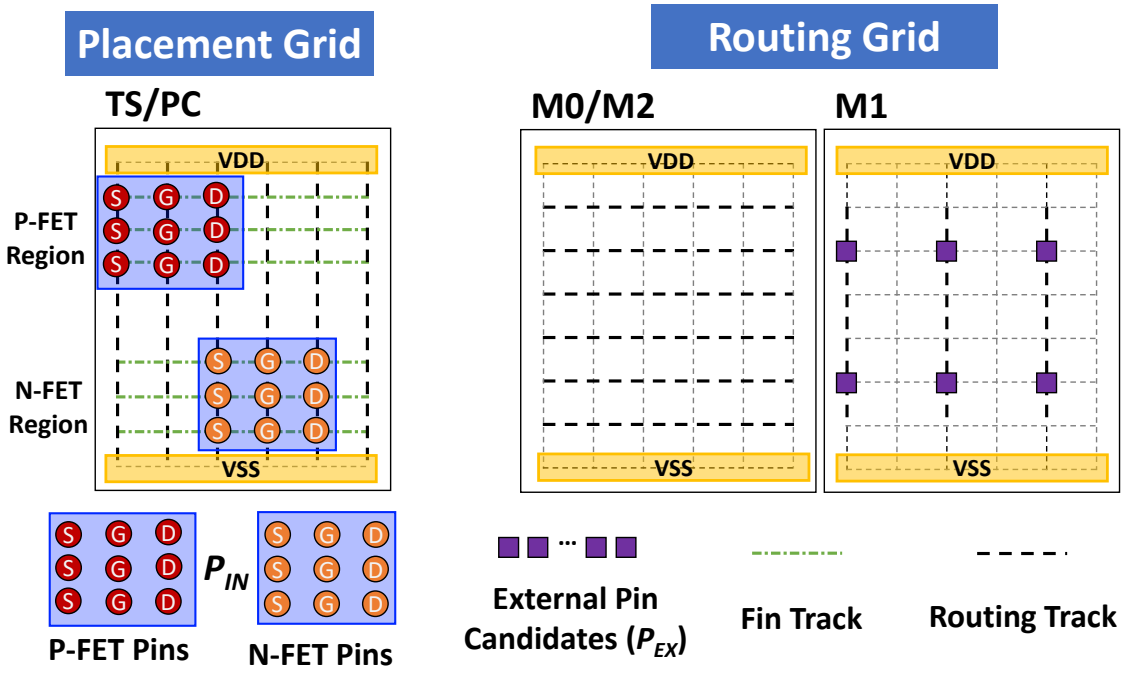


Figure 4.3: Grid-based placement & 3-D routing graph.

Layer Configuration. We define the grid-based placement and 3-D routing graph composed of four metal layers (i.e., TS/PC, $M0$, $M1$, and $M2$) as shown in Figure 4.3. In practice, routing layers' multiple interchanges on timing critical paths are undesirable due to the severe

performance losses caused by the high resistance of VIA elements. Therefore, we determine the weighted cost $w_{v,u}$ of VIA metal segments by four times higher than that of horizontal and vertical metal segments. In placement grid (i.e., TS/PC), there are three placement tracks (i.e., fin tracks) for an allocation of FETs in the corresponding P-FET/N-FET region. Due to the limited placement tracks, we only consider the single-stack placement of FETs in each region. The routing grid (i.e., $M0/1/2$) consists of six horizontal tracks.

Internal Pin (P_{IN}) for FET. P_{IN} refers to the source, drain, and gate of each FET and is defined in placement graph as depicted in Figure 4.3. The location of each pin is dynamically determined by placement formulation (Section 4.3.1) and is associated with the flow formulation for routing through our DPA (dynamic pin allocation) scheme (Section 4.3.3).

External Pin (P_{EX}) for I/O Pin Access. P_{EX} represents I/O pins of a standard cell. Vertices (depicted in purple squares in Figure 4.3) interconnected to P_{EX} on $M1$ layer are defined as candidates for each I/O pin's access point. One of these candidates is assigned as an I/O pin access point by our flow formulation. The routed metal segments on $M1$ and $M2$ layers including the assigned vertices represent the I/O pin of a standard cell for the detailed routing phase.

4.3 Simultaneous Placement & Routing

In this section, we describe our SMT formulation of the constraints for the proposed *SP&R* framework. This section consists of *(i) Transistor Placement*, *(ii) In-Cell Routing*, and *(iii) Dynamic Pin Allocation (DPA)*.

4.3.1 Transistor Placement

FET Configuration. Figure 4.4 illustrates an example of variable types of a FET with size of 3. There are four possible FET types such as "1 finger", "1 finger (flipped)", "3 fingers", and "3 fingers (flipped)". Since we only consider a single-stack placement in sub-7*nm* technology

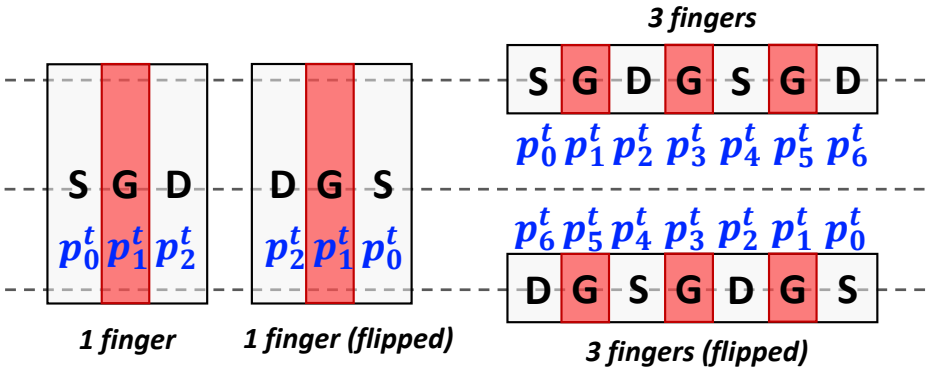


Figure 4.4: Configuration of a FET with size of 3.

nodes, *SP&R* selects FET types having the minimum number of fingers (i.e., "1 finger" and "1 finger (flipped)") to minimize the cell size. *SP&R* defines the pin information based on the selected FET type (i.e., p_0^t, p_1^t, and p_2^t as shown in "1 finger" cartoons of Figure 4.4).

Diffusion Sharing (DS). DS is a common placement technique when the net information and the diffusion height (numbers of fins in P-FETs and N-FETs) are the same between pins of individual FETs. However, inevitably, standard cells must have different diffusion heights to enable flexible power-performance exploration. Also the disparity in diffusion height brings harmful side effects such as yield loss or neighbor diffusion effect [88] due to the distortion during diffusion process for adjacent FETs. Therefore, in a conventional physical design flow, these size transition within a standard cell is captured by the library characterization because the diffusion shapes are pre-determined. *SP&R* provides an optional FET size transition (FST) in diffusion sharing (DS) between FETs with different diffusion heights, to support the library characterization in various process architectures. Figure 4.5 depicts the DS rule according to the FST option. When the FST is disabled, DS is not allowed between FETs with different diffusion heights.

Diffusion Break (DB). As shown in Figure 4.6, DB refers to the minimum space d between distinct diffusion regions when they are not shared due to the different net information or different diffusion heights. *SP&R* supports single diffusion break (SDB) and double diffusion

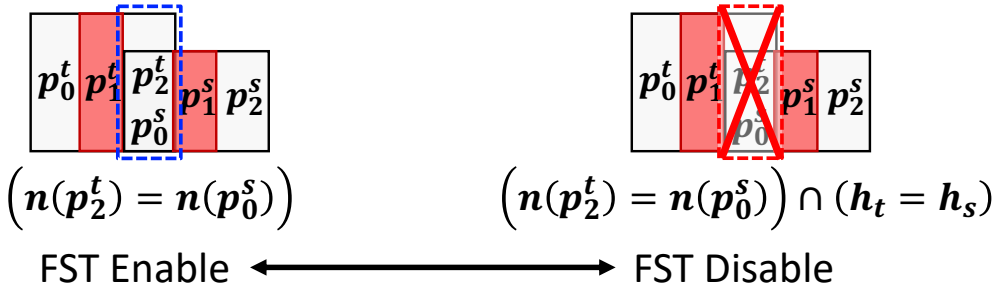


Figure 4.5: Diffusion sharing (DS) with the FET size transition (FST) option.

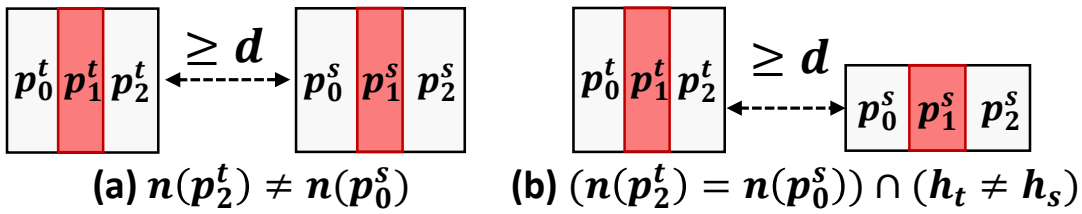


Figure 4.6: Diffusion break (DB) (a) different net information, (b) different diffusion heights with FST disable.

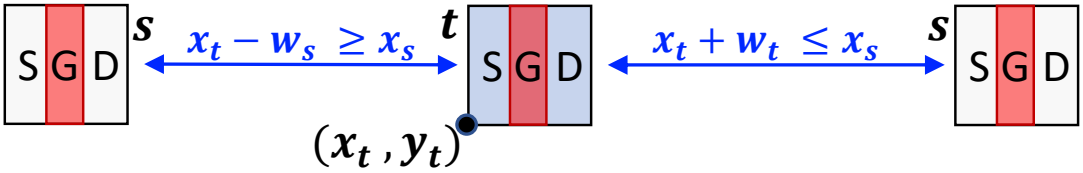


Figure 4.7: Relative positions between two FETs.

break (DDB). The minimum space d of SDB and DDB is 2 and 4, respectively.

Relative Positioning Constraint (RPC). We utilize the conventional floorplanning design approach (i.e., *Relative Positioning Constraint (RPC)*) for the transistor placement problem [89]. Since we only consider a single-stack placement, all transistor positions can be represented by two RPCs as shown in Figure 4.7 and Expression (4.6).

$$\begin{aligned} \textbf{Right:} \quad & x_t \geq x_s + w_s, \\ \textbf{Left:} \quad & x_t + w_t \leq x_s, \end{aligned} \qquad \forall t, s \in T, t \neq s \tag{4.6}$$

According to the input parameters as shown in Algorithm 2, our *SP&R* calls sub-procedures defined in Algorithms 3-5 to set corresponding RPC formulation with DS and DB. Algorithm 3 sets the RPC for SDB when SDB is a major DB. Each RPC on the left/right side is separated into two cases with/without DS. This geometric SMT constraint ensures that only one case is enabled at once and determines the position and the flip status of FETs. When DDB is a major DB, there exists a case that the other FETs are placed between two FETs of interest and share a diffusion region with one of the two FETs. This prohibits the consecutive DS occurrence of FETs because the RPC only considers the relative position between two FETs. To prevent this case, the RPC for DDB refers the DS indicators o_l^t and o_r^t as shown in Algorithm 4. When FET t is on the right side of FET s ,also when FET t is sharing a diffusion on the left side (Lines 6-8), the distance between FETs t and s is set to the minimum value 2. So the RPC does not restrict DS of FET s and the FET that is placed between FETs t and s. If there are no FETs between FETs t and s, the RPC sets the distance using d_d (Lines 9-11). Algorithm 5 represents the RPC with the mixed SDB in a crossover. Since *SP&R* minimizes the cell size, all the pairs of FETs which have the same net information on their facing nodes must be placed with DS. However, the gate signals' mismatch in a crossover prohibits DS, resulting in the "skip device". Therefore, we can detect the "skip device" by finding FET pairs that are not sharing diffusion regions (Lines 9, 23) even though they meet the sharing conditions (Lines 10, 24).

4.3.2 In-Cell Routing

We adopt conditional design rule-aware multi-commodity network flow theory to formulate the in-cell routing problem as described in [44, 5, 66, 87]. Specifically, the refined con-

Algorithm 2: 1-D Relative Positioning Constraint (FETs t, s)

Input: t, s: a pair of FETs in T, a set of FETs in a crossover: T_c
if *single diffusion break (SDB) is set* **then**
    SetRPCwithSDB (t, s); // Algorithm 3
end
else if *double diffusion break (DDB) is set* **then**
    if $t \in T_c$ & $s \in T_c$ **then**
        SetRPCwithMixedDB (t, s); // Algorithm 5
    end
    else
        SetRPCwithDDB (t, s); // Algorithm 4
    end
end

Algorithm 3: SetRPCwithSDB (FETs t, s)

Input: t, s: a pair of FETs, d_s: distance of a single diffusion break
/ FST: 0-1 indicator if FET size transition is enabled */*
// Set SMT Constraint
if $x_t > x_s + w_s$ **then**
    $x_t \geq x_s + w_s + d_s$;
end
else if $x_t = x_s + w_s$ & $n(p_l^t) = n(p_r^s)$ & *(FST | (!FST &* $h_t = h_s$*))* **then**
    $x_t = x_s + w_s$;
end
else if $x_t + w_t < x_s$ **then**
    $x_t + w_t + d_s \leq x_s$;
end
else if $x_t + w_t = x_s$ & $n(p_r^t) = n(p_l^s)$ & *(FST | (!FST &* $h_t = h_s$*))* **then**
    $x_t + w_t = x_s$;
end
else
    Unsatisfiable Condition;
end

straints for *commodity flow conservation (CFC)* and *vertex exclusiveness (VE)* in uni-directional edges [44, 87] are implemented in our framework to reduce the search space of the routing formulation. The routing formulation consists of two parts, *flow formulation* and *conditional design rules* as shown in Figure 4.8. The flow formulation secures the routing path between the

Algorithm 4: SetRPCwithDDB (FETs t, s)

Input: t, s: a pair of FETs, d_d: distance of a double diffusion break
/ leftmost (resp. rightmost) pin of t and s: p_l^t and p_l^s (resp. p_r^t and p_r^s) */*
/ FST: 0-1 indicator if FET size transition is enabled */*
/ o_l^t (or o_r^t): 0-1 indicator if FET t shares diffusion on the left (or right) side */*
// Set SMT Constraint
if $x_t > x_s + w_s$ & o_l^t **then**
    $x_t \geq x_s + w_s + 2$;
end
else if $x_t > x_s + w_s$ & $!o_l^t$ **then**
    $x_t \geq x_s + w_s + d_d$;
end
else if $x_t = x_s + w_s$ & $n(p_l^t) = n(p_r^s)$ & *(FST | (!FST &* $h_t = h_s$*))* **then**
    $x_t = x_s + w_s$;
end
else if $x_t + w_t < x_s$ & o_r^t **then**
    $x_t + w_t + 2 \leq x_s$;
end
else if $x_t + w_t < x_s$ & $!o_r^t$ **then**
    $x_t + w_t + d_d \leq x_s$;
end
else if $x_t + w_t = x_s$ & $n(p_r^t) = n(p_l^s)$ & *(FST | (!FST &* $h_t = h_s$*))* **then**
    $x_t + w_t = x_s$;
end
else
    Unsatisfiable Condition;
end

source and the sink for each commodity without heuristic modeling. The conditional design rules work as constraints to route through design-rule violation-free paths. The built-in functions such as at-most k (AMk) and at-least k (ALk) are used to formulate cardinality constraints.

For the flow formulations, *SP&R* implements flow formulations such as *Edge Assignment* and *Metal Segment* by utilizing the same methodology of [66, 87]. The refined SMT representations of *CFC* and *VE* are as follows.

Commodity Flow Conservation (CFC). Expression (4.7) represents the CFC constraint. The number of activated commodity-flow indicators $f_m^n(v,u)$ between a certain vertex v and its

Algorithm 5: SetRPCwithMixedDB (FETs t, s)

Input: t, s: a pair of FETs, d_s (or d_d): distance of a single (or double) diffusion break
/ leftmost (resp. rightmost) pin of t and s: p_l^t and p_l^s (resp. p_r^t and p_r^s) */*
/ FST: 0-1 indicator if FET size transition is enabled */*
/ o_l^t (or o_r^t): 0-1 indicator if FET t shares diffusion on the left (or right) side */*
// Set SMT Constraint
if $x_t > x_s + w_s$ & o_l^t **then**
    $x_t \geq x_s + w_s + 2$;
end
else if $x_t > x_s + w_s$ & $!o_l^t$ **then**
    if $n(p_l^t)$ *equals* $n(p_r^s)$ **then**
        $x_t \geq x_s + w_s + d_s$;
    end
    else
        $x_t \geq x_s + w_s + d_d$;
    end
end
else if $x_t = x_s + w_s$ & $n(p_l^t) = n(p_r^s)$ & *(FST | (!FST &* $h_t = h_s$*))* **then**
    $x_t = x_s + w_s$;
end
else if $x_t + w_t < x_s$ & o_r^t **then**
    $x_t + w_t + 2 \leq x_s$;
end
else if $x_t + w_t < x_s$ & $!o_r^t$ **then**
    if $n(p_r^t)$ *equals* $n(p_l^s)$ **then**
        $x_t + w_t + d_s \leq x_s$;
    end
    else
        $x_t + w_t + d_d \leq x_s$;
    end
end
else if $x_t + w_t = x_s$ & $n(p_r^t) = n(p_l^s)$ & *(FST | (!FST &* $h_t = h_s$*))* **then**
    $x_t + w_t = x_s$;
end
else
    Unsatisfiable Condition;
end

adjacent vertices $a(v)$ is 1 (Exactly-1) in case of source s^n or sink d_m^n, and is 0 or 2 in the other cases. "Exactly-k" constraints are represented by combining "**AMk**" and "**ALk**".

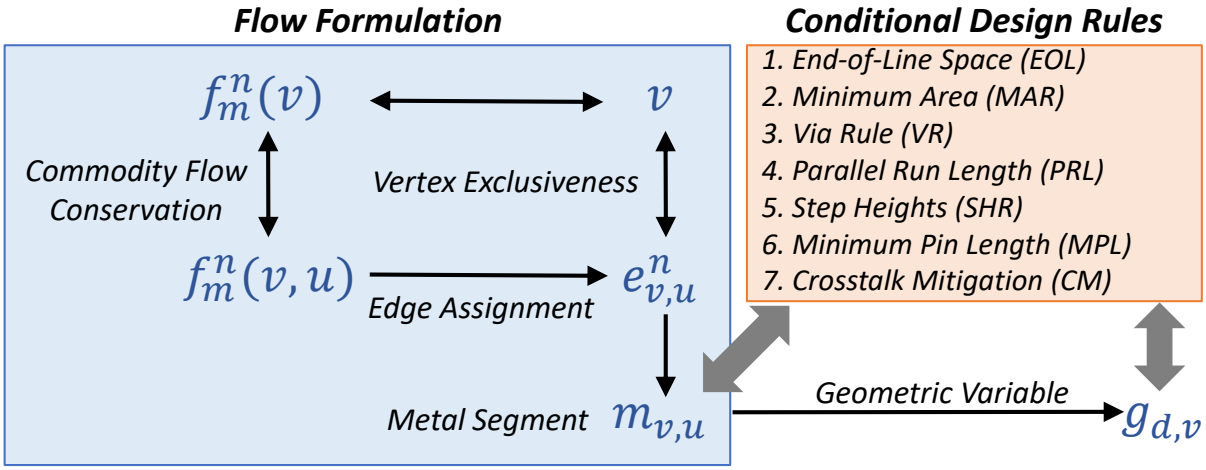


Figure 4.8: Flow formulation with conditional design rules.

$$\begin{cases} \mathbf{AL1}\left(F_m^n(v)\right) \wedge \mathbf{AM1}\left(F_m^n(v)\right), & \text{if } v = s^n, d_m^n \\ \mathbf{AM0}\left(F_m^n(v)\right) \vee \left\{\mathbf{AL2}\left(F_m^n(v)\right) \wedge \mathbf{AM2}\left(F_m^n(v)\right)\right\}, & \text{otherwise} \end{cases} \tag{4.7}$$

$$F_m^n(v) = \{f_m^n(v,u) \mid u \in a(v)\}, \forall v \in V,\ \forall n \in N,\ \forall d_m^n \in D^n$$

Vertex Exclusiveness (VE). Expression (4.8) ensures that there are no intersecting nets on any vertices except P_{EX} (see Section 4.2.4). For P_{EX}, Exactly-k (E-k) constraint is set because the supernode of external pins should be shared as many as the number of P_{EX}. (i.e., $|P_{EX}|$) When $v = P_{IN}$ or P_{EX}, only one edge indicator must be used. Otherwise, we allow multiple uses of edges against vertex v for a certain net.

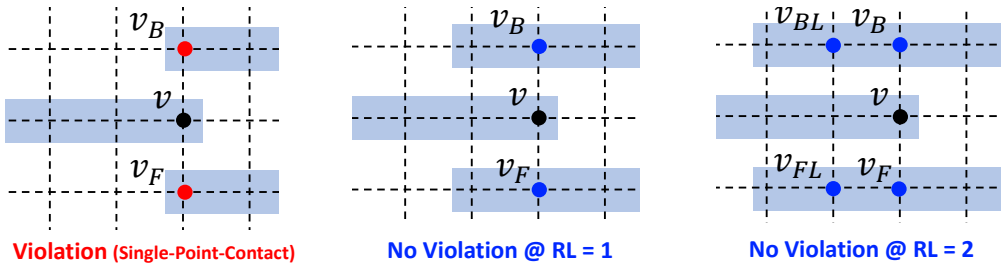


Figure 4.9: Example of PRL (Parallel Run-Length) rule.

$$\begin{cases} \mathbf{AL1}(E_{IN}(v)) \wedge \mathbf{AM1}(E_{IN}(v)), & \text{if } v = P_{IN} \\ \mathbf{ALk}(E_{EX}(v)) \wedge \mathbf{AMk}(E_{EX}(v)), k = |P_{EX}|, & \text{else if } v = P_{EX} \\ \mathbf{AM1}(\{\bigvee_{u \in a(v)} e^n_{v,u} \mid n \in N\}), & \text{otherwise} \end{cases}$$

$$E_{IN}(v) = \{e^n_{v,u} \mid u \in a(v)\}, E_{EX}(v) = \{e^n_{v,u} \mid n \in N, u \in a(v)\},$$

$$\forall n \in N, \forall v \in V \tag{4.8}$$

For the formulations of conditional design rules, previous works [5, 66, 87] mainly tackle three representative conditional design rules, e.g., *Minimum Area (MAR)*, *End-of-Line Spacing (EOL)*, and *Via Rule (VR)*. In *SP&R*, *Minimum Area* and *End-of-Line Spacing* follow the same principle of [66, 87]. Compared to [66, 87], we adopt stack via rule (stack-able) for *Via Rule* [8]. Furthermore, *SP&R* includes multi-pattern-aware design rules such as *Parallel Run Length (PRL)* and *Step Heights Rule (SHR)* [54][40]. PRL and SHR have essential roles for handling the complex line-end overlap rules of SADP/SAQP processes in advanced technology nodes. To ensure the pin-accessibility, we consider *Minimum I/O Pin Length (MPL)*.

Parallel Run-Length (PRL). PRL rule is a design rule to avoid "single-point-contact" in manufacturing SADP mask [40]. Figure 4.9 and Constraint (4.9) represent an example of PRL rule and the corresponding formulation when the run-length (RL) is 2.

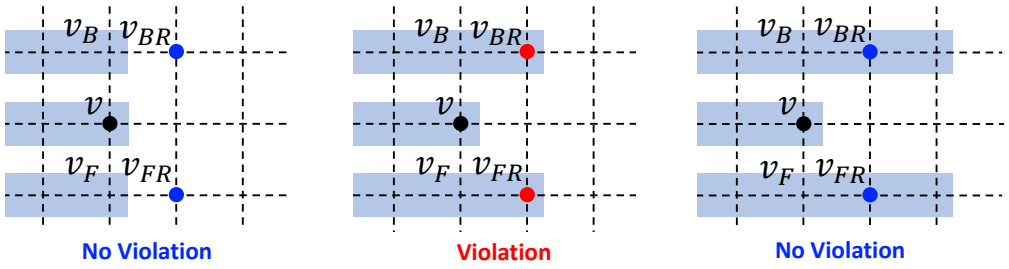


Figure 4.10: Example of SHR (Step Heights Rule).

$$\textbf{AM1}(g_{R,v}, g_{L,v_B}, g_{L,v_{BL}}); \textbf{AM1}(g_{R,v}, g_{L,v_F}, g_{L,v_{FL}}), \forall v \in V_0, V_2 \tag{4.9}$$

Step Heights Rule (SHR). SHR is a design rule to avoid "the small step" in manufacturing SADP mask [40]. Figure 4.10 and Constraint (4.10) describe an example of SHR and the corresponding formulation when the step height is 2.

$$\textbf{AM1}(g_{R,v}, g_{R,v_{BR}}); \textbf{AM1}(g_{R,v}, g_{R,v_{FR}}), \quad \forall v \in V_0, V_2 \tag{4.10}$$

Minimum Pin Length (MPL). MPL rule ensures the minimum number of metal segments of the commodity heading to the external pin P_{EX} on $M1$ layer. At-least 1 metal segment on $M1$ layer must be assigned to the commodity whose sink is P_{EX} as expressed in Constraint (5.5). Then, the metal segment on $M1$ layer is extended to have the minimum length defined by MAR.

$$\textbf{AL1}(m_{v,v_F}, m_{v,v_B}), \quad \text{if } f_m^n(v, v_D) = 1 \ , \ f_m^n(v, v_U) = 1$$
$$\forall v \in V_1, d_m^n = P_{EX} \tag{4.11}$$

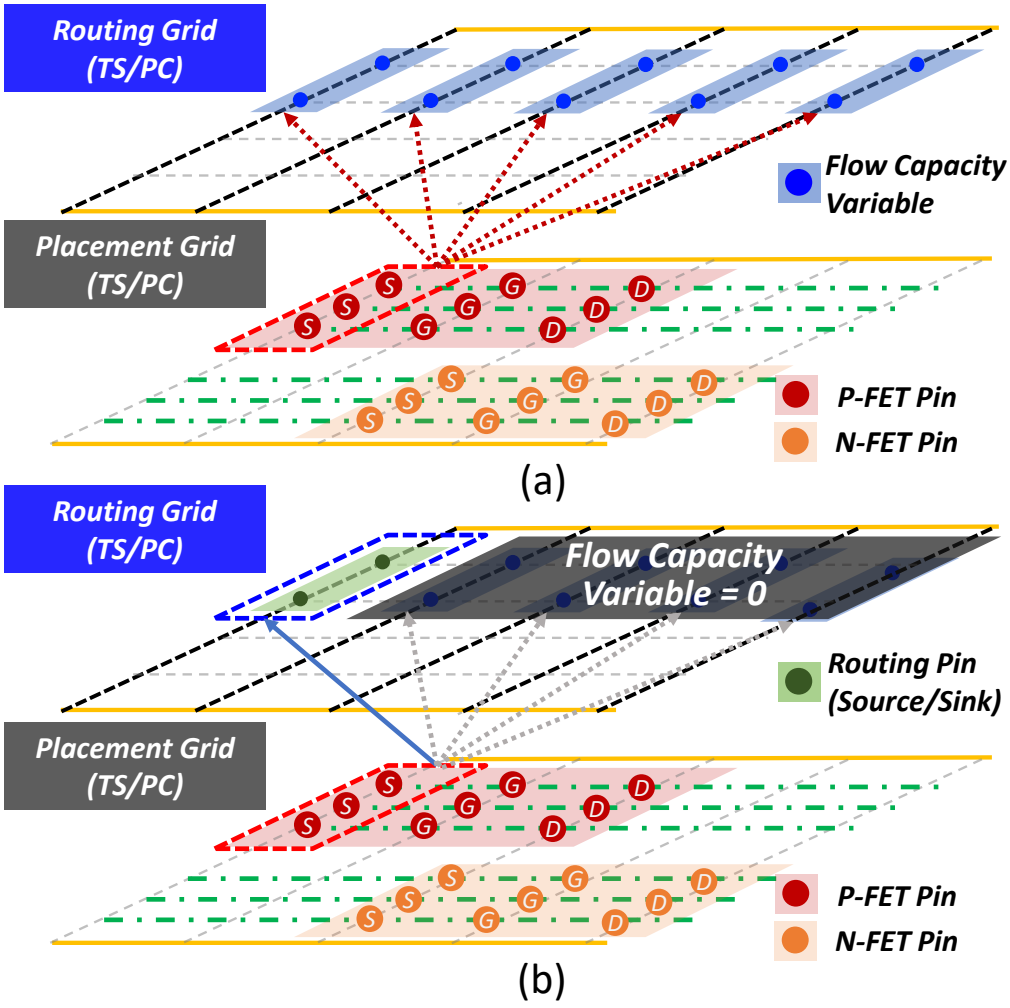


Figure 4.11: Dynamic pin allocation (DPA) between placement and routing grids.

4.3.3 Dynamic Pin Allocation (DPA)

We devise a dynamical pin allocation (DPA) scheme between placement and routing grids. In the TS/PC layer, the placement tracks are not exactly aligned with the routing tracks. Therefore, we have to map the pins of each FET on the placement grid to the routing pins on the routing grid to utilize the grid-based routing formulation as shown in Figure 4.11.

From Placement (Pin Allocation). Every pin in each FET has its own flow capacity variable $C_m^n(p,r)$ on their corresponding vertices of TS/PC routing grid as shown in Figure 4.11(a). When locations of FETs are determined by the placement formulation, the flow capacity variables

of each pin are conditionally assigned to the corresponding location of each pin. Algorithm 6 presents the flow capacity control constraint. For certain net n and commodity m, $C_m^n(p,r)$ is set as 0 if vertex r is not in the range of p. Figure 4.11(b) shows the flow capacity variables assigned to 0 outside the corresponding column of a source pin on P-FET (depicted in red dashed box).

Algorithm 6: Flow Capacity Control Constraint ($C_m^n(p,r)$)

```
/* x coordinate (resp. y coordinate) of a routing grid r: x_r (resp. y_r) */
/* Height and x coordinate (resp. y coordinate) of a pin p: h_p and x_p (resp. y_p) */
/* Single column pin only: Set of x is singleton */
/* p is either source or sink of a net n and commodity m */
// Set SMT Constraint
if (x_r ≠ x_p) | (y_r < y_p) | (y_r > y_p + h_p) then
    C_m^n(p,r) = 0;
end
else
    C_m^n(p,r) is Determined by Routing Formulation;
end
```

To elaborate further with a running example in Figure 4.12, each internal pin (i.e., source/gate/drain) of Figure 4.12(a), (b), and (c) has only two feasible vertices (blue circles) for pin access through the flow capacity control algorithm. All feasible sets of vertex r in TS/PC layer can be either the source s^n or sink d_m^n in the routing formulation by connecting the flow capacity variable $C_m^n(p,r)$ to flow variable $f_m^n(v,u)$ as follow.

To Routing (Flow Capacity Connection). The flow variable $f_m^n(v,u)$ (in Expression (4.7) of routing formulation) is associated with the flow capacity variable $C_m^n(p,r)$ by the constraint described in Algorithm 7. Each $f_m^n(v,u)$ is determined by the routing formulation when vertex v is the internal pin p, and the adjacent vertex u is the adjacent vertex r of p in TS/PC (i.e., V_0). Thus, routing formulation can recognize the feasible sets of r in V_0 layer as routing pins (depicted in blue dashed box of Figure 4.11(b)).

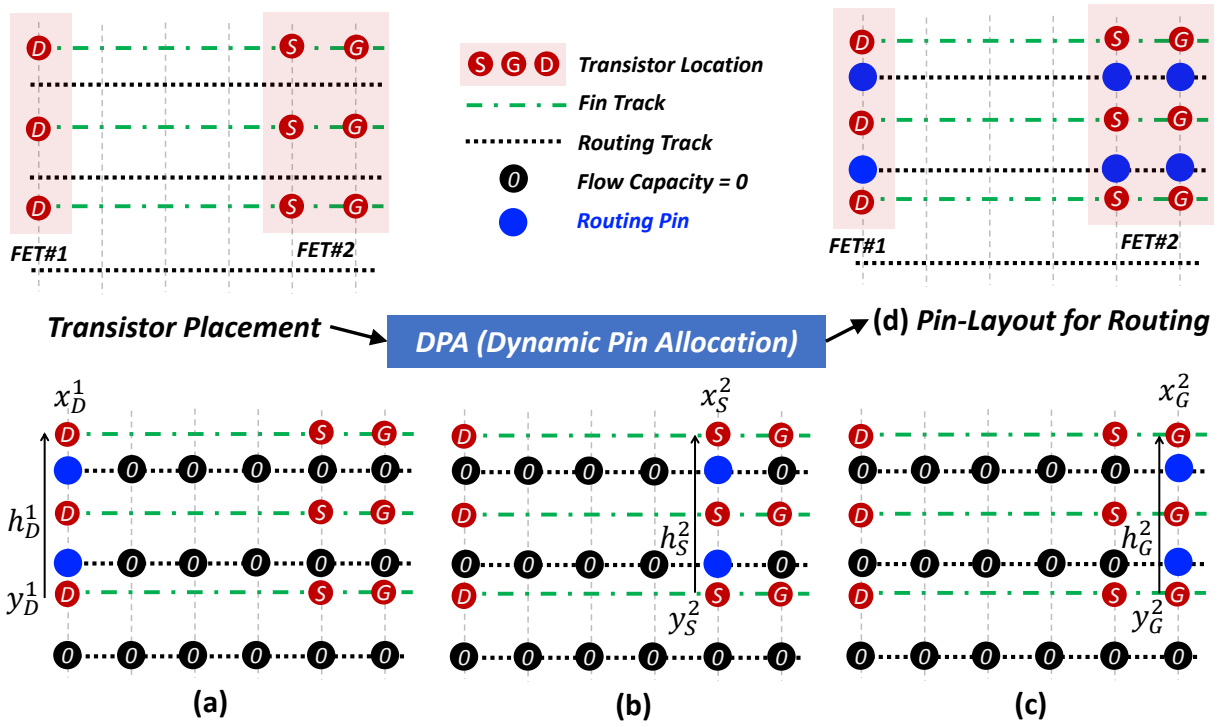


Figure 4.12: An example of pin allocation through DPA. Capacity control for (a) drain pin of FET#1, (b) source pin of FET#2, and (c) gate pin of FET#2. (d) Selected pin-layout for routing.

Algorithm 7: Flow Capacity Connection $(f_m^n(v,u), C_m^n(p,r))$

/* *p is either source or sink of a net n and commodity m* */
/* $\forall r \in a(p),\ \forall r \in V_0$ */
// Set SMT Constraint
if $C_m^n(p,r) = 0$ **then**
  $f_m^n(v = p, u = r) = 0$;
end
else
  $f_m^n(v = p, u = r)$ is Determined by Routing Formulation;
end

4.4 Experiments

The proposed SP&R (Simultaneous Place-and-Route) framework is implemented in *Perl/SMT-LIB* 2.0 standard-based formula and validated on a Linux workstation with 2.4GHz Intel Xeon E5-2620 CPU and 256GB memory. The multi-threaded SMT Solver *Z3* (version

4.8.5) is used to produce the optimized solution through the proposed SP&R formulation. SP&R generates the "design layout" file including the information for the transistors (i.e., FETs), nets (i.e., target nets for in-cell routing), and I/O pins (i.e., P_{EX}) by using schematic information of cell logic.

4.4.1 Sequential vs. Simultaneous P&R

Since *SP&R* simultaneously searches the solution in the combined search space of placement and routing, the result of *SP&R* is the optimal solution with respect to the multiple objectives (i.e., cell size, #$M2$ track, and total metal length) in the lexicographic order. We compare our simultaneous P&R approach to the conventional sequential approach [8]. To simulate the sequential P&R of [8], we first find a placement solution with the minimum cell size satisfying the flow formulation described in Section 4.3. Then we obtain the optimized routing solution with respect to the total metal length satisfying all routability-related formulation (i.e., flow formulation and design rule constraints). Figure 4.13 shows the difference in terms of the key metrics between *SP&R* and the sequential approach for a HA_X1 cell. With the same cell size of 10 CPP (contact poly pitch), *SP&R* (Figure 4.13(b)) respectively reduces the total metal length and #$M2$ tracks by 4.6% (241 $\rightarrow$ 230) and 50% (2 $\rightarrow$ 1) by virtue of the DPA scheme compared to the sequential approach (Figure 4.13(a)).

4.4.2 Optimization for DFM and I/O pin accessibility

SP&R applies strict design-rule constraints for DFM-aware cell-layout design automation. Figure 4.14 shows an example of a generated AOI22_X1 cell layout satisfying all pre-discussed design-rule constraints[1]. The metal segments (a) and (b) (red dashed region) are extended to satisfy SHR and MAR, respectively. The blue dashed region shows that the metal segment is

[1]In this research, *SP&R* considers several representative design rules. By the nature of on-grid routing, the authors firmly believe that all other conditional design rules can be properly formulated and integrated.

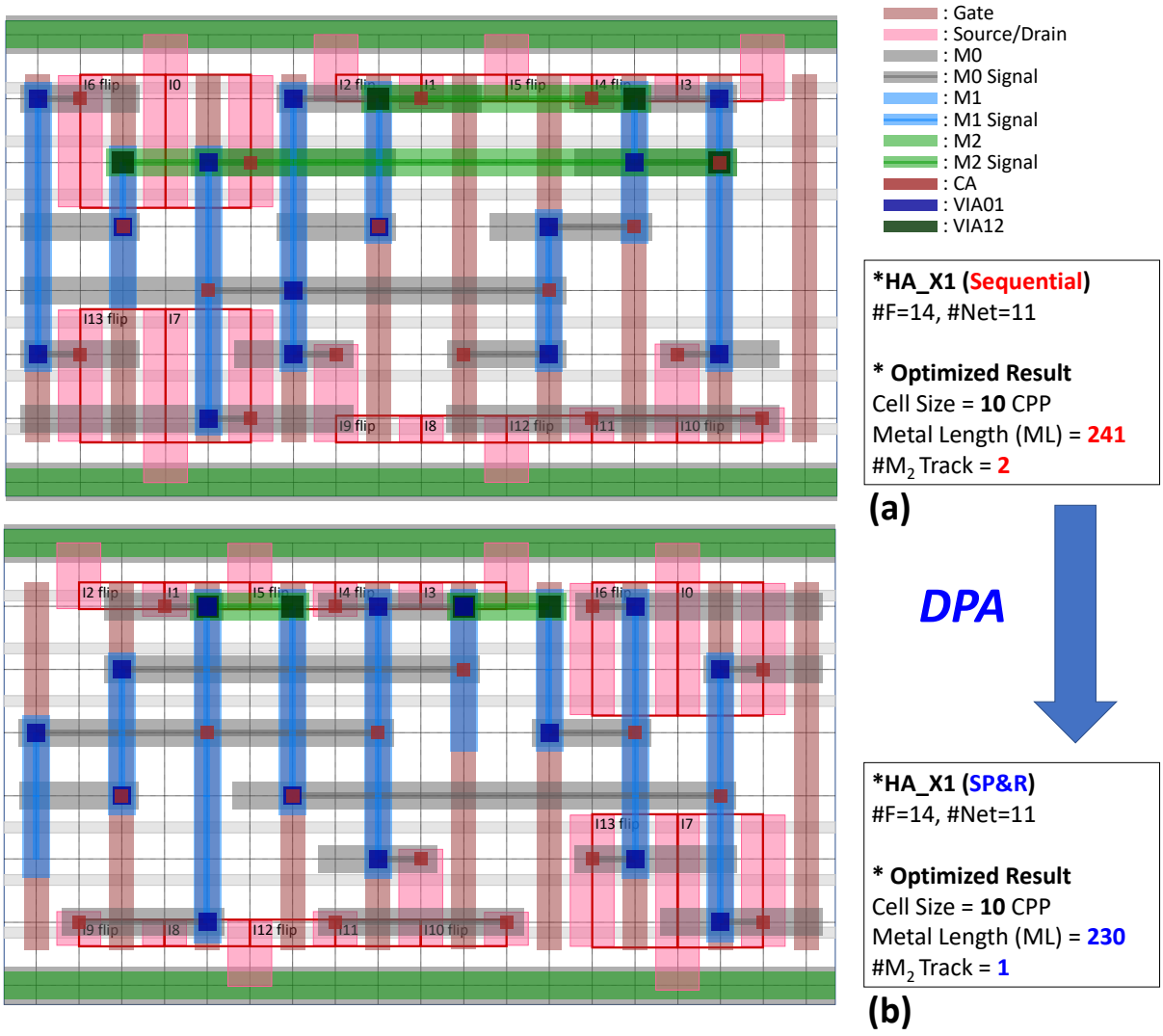


Figure 4.13: The comparison between (a) sequential (ML=241, #$M2$=2) and (b) simultaneous P&R (the proposed *SP&R*) (ML=230, #$M2$=1) using HA_X1.

extended to satisfy MPL design rule for I/O pin accessibility.

4.4.3 Experimental Statistics

Table 4.1 presents our experimental results with respect to SMT formulation, cell layout optimization for 9 standard cells, and reference data of [8] for the comparison. As shown in Table 4.1, SP&R reduces the metal length by 10.5% (18.2%) on average (best case) compared to sequential P&R. Moreover, SP&R reduces the number of consumed $M2$ tracks from 1.3 to 0.8

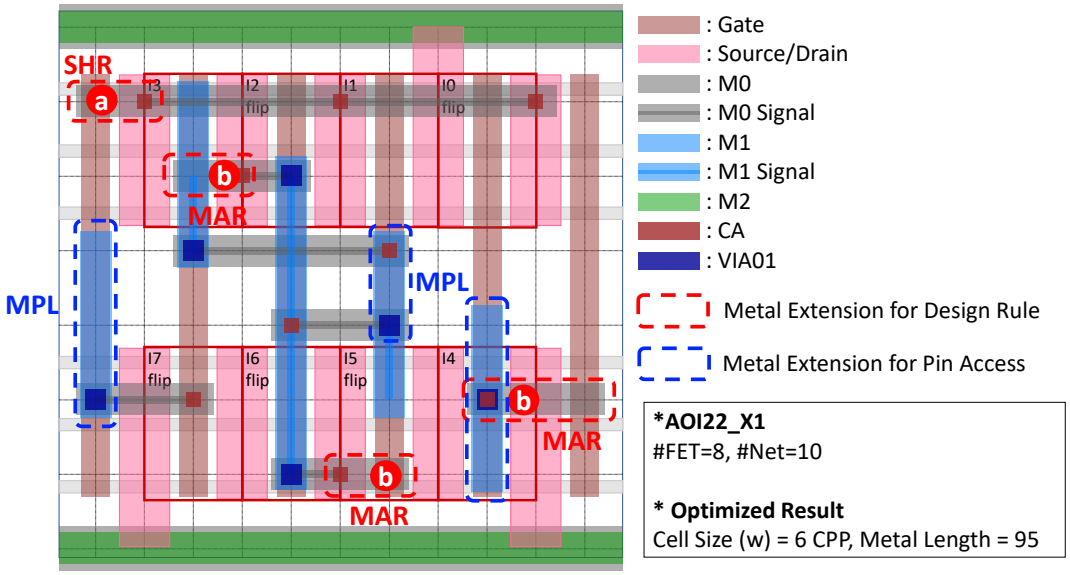


Figure 4.14: An optimized cell layout for DFM and I/O pin accessibility.

on average. SP&R obtains the optimal cell layout within 7 minutes for 8 out of 9 standard cells. For LHQ_X1 cell, the optimal cell layout is found within 24 minutes (T_O). In SMT formulation, the number of variables and constraints with respect to P&R dominantly relies on the number of FETs and nets. The overhead of variables and constraints for DPA is about 3.7%, 1.0% on average of P&R variables and constraints, respectively. With small overhead, DPA successfully combines the placement and routing. The reference data of [8] represents the cell specification (i.e., #FET and #Net) and P&R result (i.e., w and T_D) of cells which have similar complexity with our standard cell examples. This demonstrates that our SP&R produces the optimal cell layout solution with reasonable time overhead.

4.5 Conclusion

In this chapter, we have described our simultaneous placement and routing framework, i.e., SP&R , a new standard cell design automation framework with the simultaneous place-and-route

[2] All experiments of [8] were executed on a single-threaded 2.20GHz Intel Xeon E5-2699 v4 machine and using CPLEX 12.6.

Table 4.1: Experimental Statistics: P&R = number of variables and constraints for placement and routing respectively, DPA = number of variable/constraint for DPA(dynamic pin allocation), w = cell width in tracks, ML = total metal length, M_2 = number of used M_2 tracks, ML Impr. = ($ML_{sequential}$ - $ML_{simultaneous}$) / $ML_{sequential}$, T_O = execution time of SP&R, T_D = total P&R execution time with optimal manufacturability of [8] (i.e., $t3 + t5$ of [8]). All time is in [mm:ss].

Cell Specification			SMT Formulation				Cell Layout Optimization								[8] (Sequential/ILP^2)				
Name	#FET	#Net	#Variable		#Constraint		Sequential P&R			Simultaneous P&R				ML					
			P&R	DPA	P&R	DPA	w	ML	M_2	w	ML	M_2	T_O	Impr.[%]	#	#FET	#Net	w	T_D
AOI22_X1	8	10	10,268	314	17,908	145	7	104	0	7	95	0	00:31	8.7	1	8	11	6	00:06
NAND4_X1	8	10	9,885	294	17,160	135	7	79	0	7	79	0	02:06	-	2	8	10	6	00:08
NAND4_X2	8	10	35,770	1,424	65,766	672	14	292	1	14	239	0	06:25	18.2	3	8	11	12	00:29
AOI22_X2	8	10	38,638	1,538	72,660	723	15	308	1	15	303	1	02:53	1.6	4	8	11	12	00:52
OAI22_X2	8	10	38,638	1,538	72,660	723	15	355	2	15	303	1	03:46	14.6	5	14	16	12	05:07
XOR2_X1	10	9	13,110	476	23,008	222	8	125	0	8	125	0	00:20	-	6	8	11	16	06:19
MUX2_X1	12	12	20,044	674	33,770	320	10	253	1	10	253	1	01:42	-	7	17	22	12	02:50
HA_X1	14	11	25,141	950	43,094	450	11	308	2	11	271	1	01:29	12.0	8	11	15	12	00:56
LHQ_X1	16	17	41,386	1,524	74,986	742	15	658	5	15	553	3	23:50	16.0	9	8	11	16	03:12
Average	10.2	11.0	22,613.6	838.0	40,624.9	395.6	11.3	275.8	1.3	**11.3**	**246.8**	**0.8**	**04:47**	**10.5**		10.0	13.1	11.6	02:13

(i.e., DPA) scheme. SP&R provides fully automated optimal solutions for standard cell layout design through exploring the combined search space between placement and routing. We validate that the proposed SP&R is capable of generating the cell layout through the OMT feature of SMT solver for the practical standard cell examples, successfully avoiding DRVs.

In the next chapter, we introduce orchestrated tactics to improve the scalability of standard cell synthesis automation, resulting in the generation of a whole set of a standard cell library in sub-7*nm*. And we suggest standard cell scaling framework which is an important parametric DTCO study in advanced technology nodes (i.e., sub-5*nm*).

This chapter contains materials from following publications: “SP&R: Simultaneous Placement and Routing framework for standard cell synthesis in sub-7nm”, by Dongwon Park, Daeyeal Lee, Ilgweon Kang, Sicun Gao, Bill Lin, and Chung-Kuan Cheng, which appears in IEEE/ACM Asia and South Pacific Design Automation Conference, January 2020. “SP&R: SMT-based Simultaneous Place-&-Route for Standard Cell Synthesis of Advanced Nodes”, by Daeyeal Lee, Dongwon Park, Chiatung Ho, Ilgweon Kang, Hayoung Kim, Sicun Gao, Bill Lin, and Chung-Kuan Cheng, which appears in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, November 2020. The dissertation author was the primary

investigator and author of these papers.

Chapter 5

Parametric DTCO through whole set of Standard Cell Library

5.1 Introduction

In Chapter 4, we discussed a new standard cell synthesis framework with the simultaneous place-and-route through the novel dynamic pin allocation (DPA) scheme. In this chapter, we introduce scalability improvement methods including breaking symmetry to ensure the practical scalability of proposed standard cell synthesis framework. With a whole set of a standard cell library, generated by our enhanced framework, we propose the standard cell scaling framework which is the crucial exploration in DTCO, when we generated the scaled IC design in advanced technology nodes such as sub-5*nm*.

With the scaling of VLSI technologies, we encounter smaller and denser cells and more complicated design rules, which call for a framework of electronic design automation tool suite to manage the complexity of the standard cell designs [90]. As shown in Figure 1.2, standard cell height has been reduced from 9 metal tracks at 20*nm* node in 2012 to 6 tracks at 7*nm* node in 2019 and is predicted to be 5 tracks or even below for 3*nm* node in the near future. The track number decrements causes pin-accessibility and routability problems. In the meantime, the design rules have grown in the quantity and complexity as the scaled dimension reaches much below photonic wavelength. The conversion of extreme ultraviolet lithography (EUV) alleviates some constraints but creates other design restrictions [91]. Therefore, there is a need to manage the complexity of the design rules and provide *design and technology co-optimization* (i.e. DTCO in Figure 1.2) [92].

The exploration of the scaling relies on automatic layout synthesis and conditional design rule analysis. In [81, 82, 8, 83], the authors reported full automation of standard cell layout. However, the placement and routing are performed in separate operations. Recently, in [44], Park et al. utilized satisfiability modulo theories (SMT) to integrate the placement and routing with a dynamic pin allocation interface and demonstrated the feasibility of the optimal layout solution. However, even if the authors demonstrate the feasibility of the framework, there are still rooms to

further improve, e.g., the scalability to deal with a whole set of practical standard cell library [3].

One of the most difficult design features of standard cell layout conversion is the pin-accessibility of the cell due to the limited number of tracks and complicated design rules. Conventional approaches for pin-accessibility in standard-cell design [37, 93] define the metric for pin-accessibility in their objectives for optimization. In this paper, we devise the constraints for pin-accessibility to guarantee the minimum pin openings through the boolean constraint (i.e., At-Least-k).

Our work consists of the following distinct features.

- Optimal placement and routing solution with automatic column insertion to minimize the cell area and ensure the pin-accessibility: We adopt the SMT with dynamic pin allocation [44] to ensure the optimality of the solution. Therefore, the users can trust the results, which are not relevant to the choice of the placement and routing heuristics. Furthermore, the results guarantee the pin-accessibility with pin openings no smaller than a thereshold.

- We develop efficient search-space reduction techniques to improve the scalability of our framework, e.g., breaking design symmetry, resulting in an average of 20.8$\times$ runtime improvement over that of reported in [44] with less than 0.2% degradation in total metal length.

- We demonstrate that our framework *SP&R* successfully generates a whole set of 7*nm* standard cell library [3] with the improved scalability and design features.

- The layout synthesis tools that observe the conditional design rules. The layout results are free from rule violations: We adopt the satisfiability approach [66] to formulate the design rules and identify the rules that impede the area reduction. We allow the tuning of the parameters so that the designers can check the tradeoffs (i.e., a hint of DTCO) between the rule adjustment and the cell area reduction.

- Standard cell architecture exploration using track number changes and design rule selections: We construct the framework and explore the standard cell scaling. We demonstrate that the devised framework can automatically layout a set of standard cells, test the architectures of various track numbers, and assess the impact of design rule parameter modifications.

The remaining sections are organized as follows. Section 5.2 describes several search-space reduction methods to improve the scalability of our framework. Section 5.4 describes our standard-cell scaling framework. Section 5.5 discusses our experiments. Section 5.6 concludes the paper.

5.2 Scalability Improvement of Standard Cell Automation

In this section, we propose search-space reduction methods to improve the scalability of the proposed *SP&R* framework. This section consists of *(i) Constraints for Practical Design Features (ii) Breaking Design Symmetry*, *(iii) Conditional Assignment*, *(iv) Localization of the Routing Region*, and *(v) Cell Partitioning*.

5.2.1 Constraints for Practical Design Features

We develop practical cell-design features to further optimize cell size (i.e., single diffusion break with a crossover) and to guarantee the stable operations of timing-critical cells (i.e., crosstalk mitigation).

Single Diffusion Break (SDB) with a Crossover. A crossover of signals in a standard cell causes "skip device" (i.e., whitespace without FETs) due to the mismatches of gate signal connections. Figure 5.1(a) shows the example of a cell placement when "skip device" occurred by the crossover of CLKN and CLKB signals. When DDB (double diffusion break) is a major diffusion break, these skip devices significantly increase cell size. In practice, to minimize the cell-area loss, SDB is used in a specific crossover region. *SP&R* provides the use of SDB for the

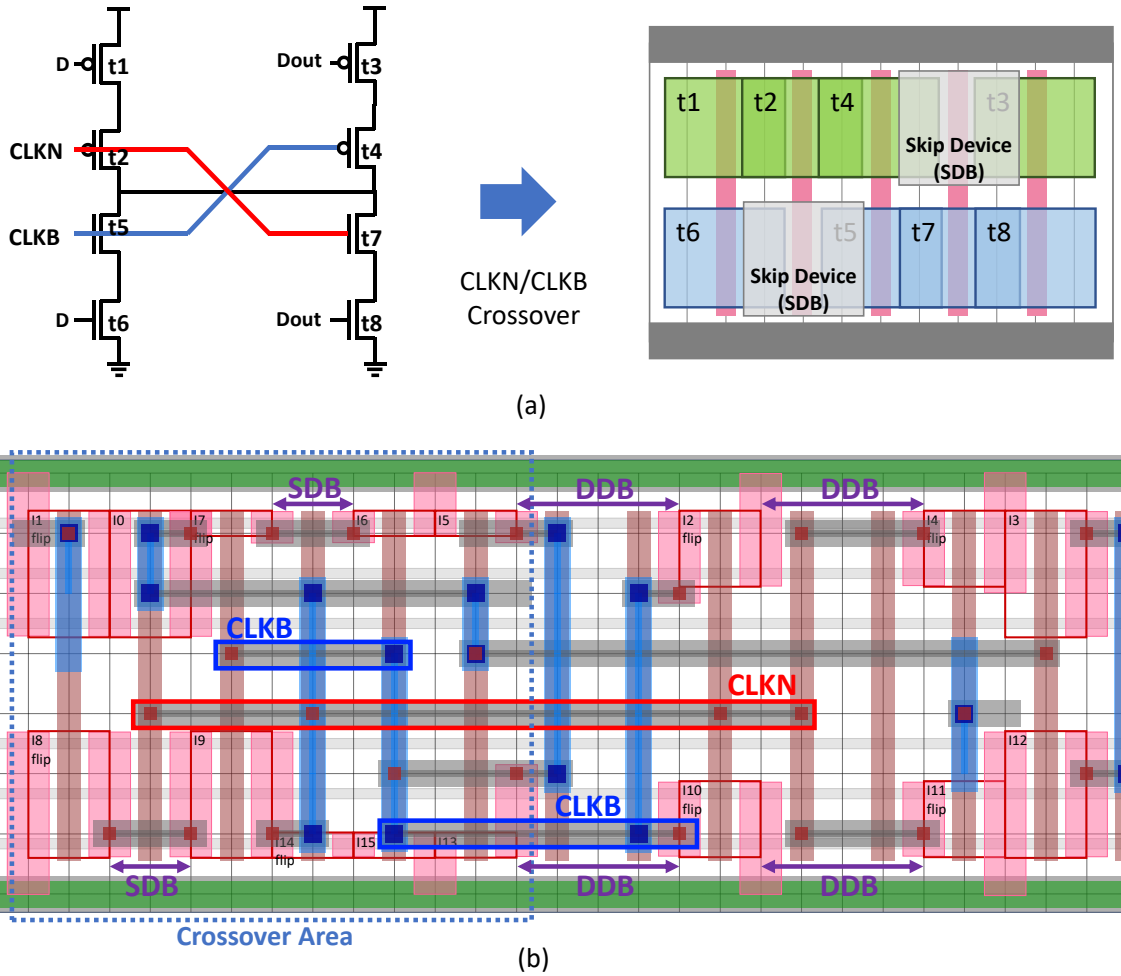


Figure 5.1: Example of SDBs. (a) a crossover area (CLKN/CLKB), (b) the crossover area in DHLx1 layout.

FETs that are specified as being in a crossover region when the major DB is DDB. Figure 5.1(b) shows an example of the generated layout (DHLx1) which is including SDB in crossover region under DDB option of DB.

Crosstalk Mitigation (CM). With evolving process technologies to sub-7*nm*, signal integrity strongly influences the performance of integrated circuits. Especially, the crosstalk between differential clock signals in the sequential cells such as latches and flipflops may cause severe timing violation thus failure of timing closure due to the cross-coupling capacitance. When the switching windows of the clock and the inverted clock overlap and switch in opposite

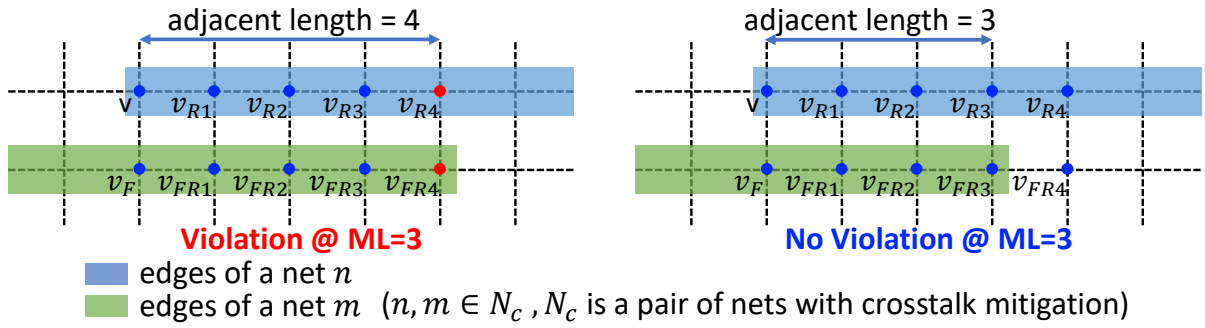


Figure 5.2: Example of crosstalk mitigation (CM) rule.

directions, the crosstalk will increase the delay of the clock nets, which may result in setup violations. More specifically, the strength of crosstalk is a function of the geometrical adjacent length (parallel running length) between adjacent nets [94]. Therefore, to mitigate the crosstalk effects for timing-critical cells, *SP&R* provides an optional design rule constraint to restrict the maximum adjacent length (*ML*) of a selected pair of nets. Figure 5.2 and Expression (5.1) represent the crosstalk mitigation constraint between nets n and m that are in a pair of nets with crosstalk mitigation N_c when $ML = 3$.

$$\begin{aligned}\mathbf{AL1}\big((\neg e^n_{v,v_{R1}} \vee \neg e^m_{v_F,v_{FR1}}),\ (\neg e^n_{v_{R1},v_{R2}} \vee \neg e^m_{v_{FR1},v_{FR2}}),\\ (\neg e^n_{v_{R2},v_{R3}} \vee \neg e^m_{v_{FR2},v_{FR3}}),\ (\neg e^n_{v_{R3},v_{R4}} \vee \neg e^m_{v_{FR3},v_{FR4}})\big)\\ \forall n,m \in N_c \qquad \forall v \in V_2\end{aligned} \tag{5.1}$$

Fig. 5.3(a) shows a layout of a DFFHQNx1 from ASAP7 library, displayed by a commercial tool [4]. The clock signals (*CLKN* and *CLKB*) with the opposite directions are routed in the adjacent *M*2 tracks with a parallel run-length over 7 CPPs. This may cause a substantial crosstalk between the clock signals which increases the delay and the power consumption. On the other hand, the result of *SP&R* with a crosstalk mitigation parameter $ML = 4$ between the clock signals successfully prevents the crosstalk by restricting the parallel run-length of those signals less than

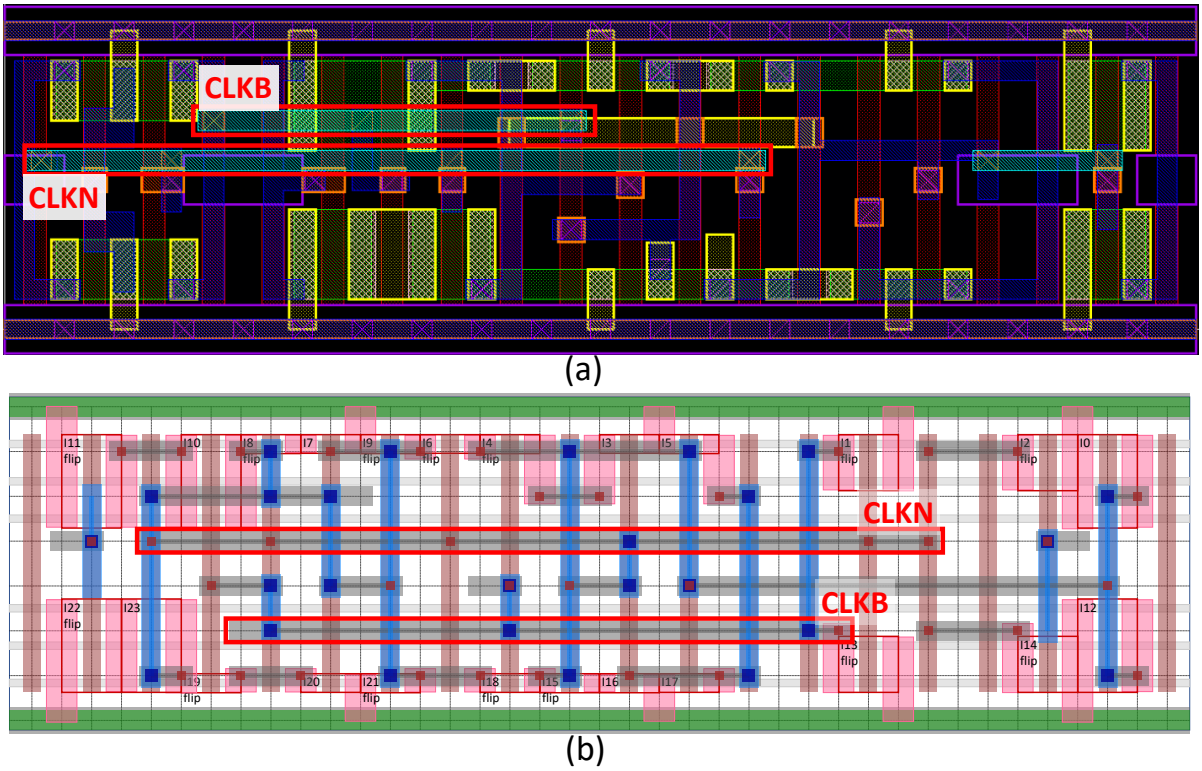


Figure 5.3: Layout of DFFHQNx1. (a) Layout from the standard cell library [3], displayed by a commercial tool [4]. (b) *SP&R*'s layout generation.

2 CPPs as shown in Fig. 5.3(b).

5.2.2 Breaking Design Symmetry

Many algorithmic applications such as search, synthesis, and optimization contain symmetries. Therefore, identifying and breaking the symmetries are crucial to improve scalability. Many SAT-based prior works have studied to reduce the search space [34, 35]. In the proposed *SP&R*, we reduce the search space by eliminating symmetries existing in standard cell layout design.

Flipping of Even-Numbered Multi-Finger FETs. Since FETs with even-numbered fingers have the same source/drain node on the leftmost/rightmost nodes, flipped FETs are the same with un-flipped FETs as shown in Figure 5.4. Therefore, for every FET t with even-

numbered fingers, ff_t is set to 0 to remove the flipped FETs from the search space.

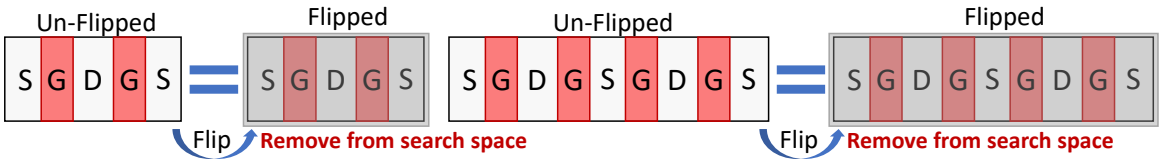


Figure 5.4: Flipped case exclusion of even-numbered finger FETs.

Flipping of Whole Cell Design In *SP&R*, every generated layout solution has a pair of dual solutions that are equivalent to their horizontal-flipped shapes as shown in Figure 5.5. The pair of dual solutions have the identical key metrics, however, they are completely different solutions in terms of the value of variables. Therefore, excluding the exploration of the dual solutions effectively cuts the search space in half. Furthermore, since *SP&R* combines the placement and the routing formulations together, and the placement of P-FETs and N-FETs are mutually dependent on each other, the dual solutions can be removed from the search space by simply setting the relative positions of P-FETs (i.e., transistor ordering) in the way of preventing the opposite order.

5.2.3 Conditional Assignment

The conditional assignment dynamically cuts the search space by assigning $true/false$ to the variables according to the intermediate conditions satisfied during the problem solving. Some routing variables depend on the assignments of other variables as shown in Figure 5.6. When a source (s^n) and a sink (d_m^n) nodes of m_{th} commodity in net n on the gate of each P-FET and N-FET are connected through PC (i.e., the x-coordinates of the source and sink are the same), the other edge variables in f_m^n outside this column will eventually be determined as 0 by the flow formulation (Figure 5.6(a)). However, if these variables are set to 0 and 1 for the variables outside of and on the column, respectively, particularly when the source and sink nodes are placed at the same column, it will reduce the searching time for determining the values by the flow formulation.

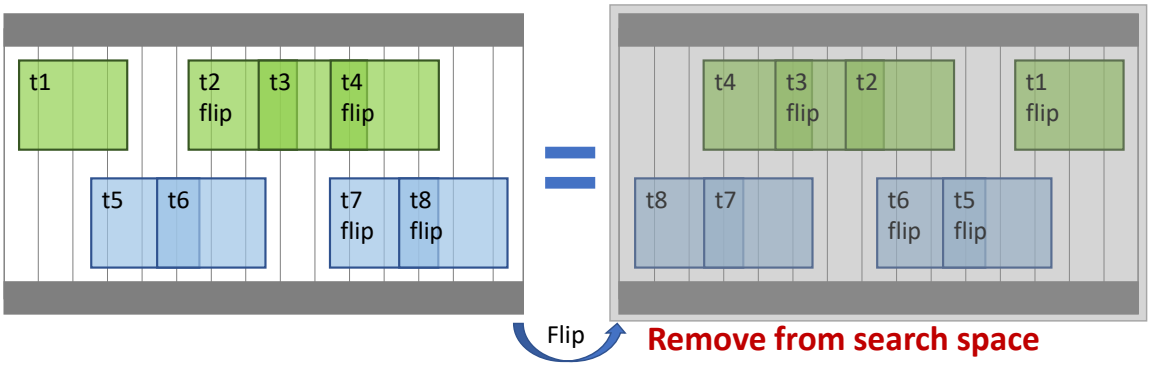


Figure 5.5: Flipped case exclusion of whole cell design from search space.

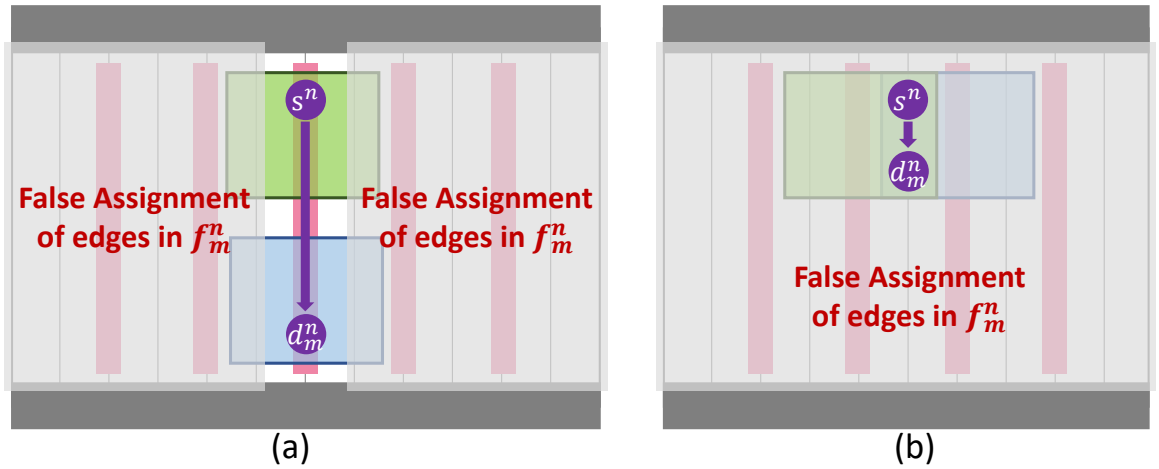


Figure 5.6: Conditional assignment. (a) A commodity flow through the same gate column, and (b) a commodity flow through the DS.

Figure 5.6(b) shows a commodity flow through TS at the same column by DS. Since the source and sink can be enabled on the same vertex, all edge variables in f_m^n are conditionally set to 0.

5.2.4 Localization of the Routing Region

The range of potential routing region for each commodity covers the entire bounding box of the cell because the location of each source/sink node is dynamically determined in *SP&R*. Therefore, a proper localization of routing regions reduces the complexity of *SP&R*.

Conditional Localization. Under the assumption that there are no detouring paths in the optimal-route solution of source-sink connectivity, the conditional localization allows to

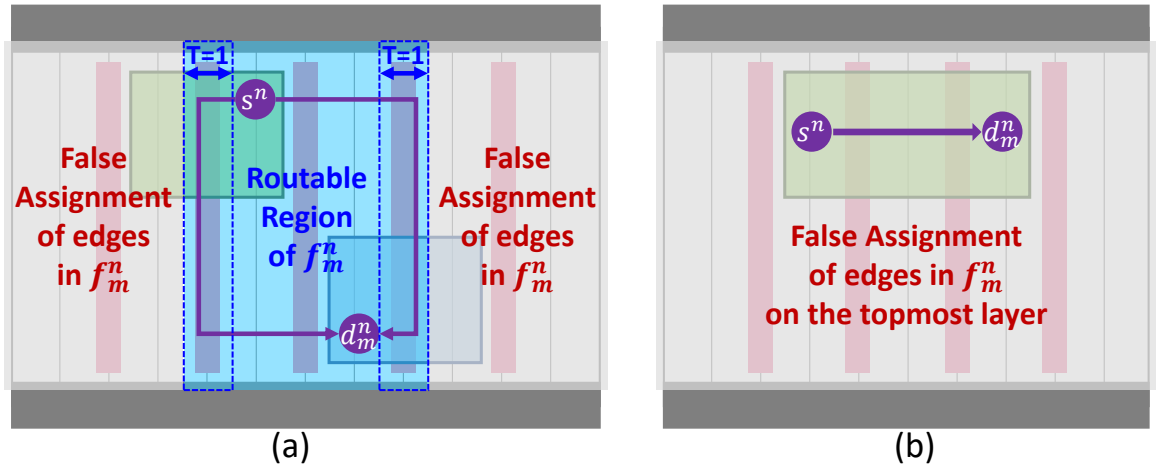


Figure 5.7: Conditional Localization. (a) Localization of commodity flows with a tolerance T=1. (b) Localization of a commodity flow within the same FET.

focus on the optimal-route reachable regions of each commodity f_m^n. Figure 5.7(a) shows the example of the conditional localization. When intermediate locations of source s^n and sink d_m^n of commodity f_m^n are determined, the optimal-route path connecting s^n and d_m^n is supposed to be in the minimum bounding box that covers both s^n and d_m^n (depicted in blue rectangle) with high probability. Therefore, the conditional $false$ assignment of the edge variables in f_m^n discards outside the localized routing region and reduces the searching time of the flow formulation. The offset with tolerance T gives a margin to prevent from over-cutting.

Localization of Intra-FET Routing. Achieving the minimum wire-length without using the topmost layer $M2$ is highly preferred for connecting nodes within the same FET. Therefore, the edge variables on the topmost layer of the commodities whose source/sink nodes are in the same FET are set to $false$ as shown in Figure 5.7(b).

5.2.5 Cell Partitioning

Since sequential cells are critically responsible for IC's performance, designing sequential cells requires special attention to timing-critical paths. Substantially, functional modules are strictly ordered by the sequential datapath to optimize the cells' PPAC. For example, flipflop's

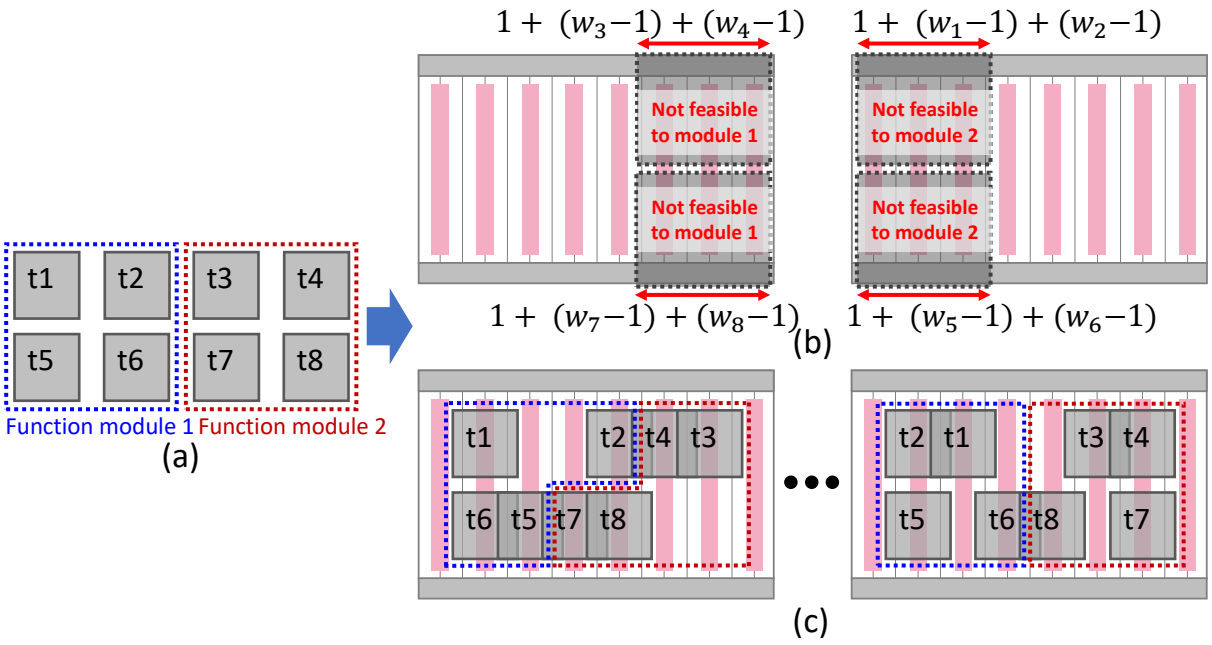


Figure 5.8: Cell Partitioning. (a) Functional module partitioning. (b) Localization of the placement area. (c) Examples of *SP&R* with cell partitioning.

functional modules i.e., Clk, Din, Dout, Master/Slave latches should follow the order of Din-Master-Slave-Dout (or Dout-Slave-Master-Din) to optimize the setup time (i.e., t_{setup}) and the delay of the flipflop (i.e., $t_{clk-to-q}$). Also, a datapath-aware placement of functional modules reduces the probability of path-level timing violations due to the twisted routing paths. Clk module is usually placed inside the cell to prevent noises from adjacent cells. To fulfill this timing-design requirement, *SP&R* performs a functional module-based cell partitioning as shown in Figure 5.8. With the pre-defined FET sets (i.e., groups by the functionality) as shown in Figure 5.8(a), *SP&R* honors the order among functional-module groups (Figure 5.8(c)). The freedom of the FET placement in each group and the DS between groups is not restricted by the DPA. Besides, ordering FET groups significantly reduces the search space by setting the relative position between FETs as well as the upper-bound and lower-bound of each FET according to the order of groups as shown in Figure 5.8(b).

5.3 Experiments for Scalability Improvement

We have implemented the proposed *SP&R* framework in *Perl/SMT-LIB* 2.0 standard-based formula and validated on a Linux desktop with 3.6GHz AMD Ryzen 5 3600 CPU and 32GB memory. The single-threaded SMT Solver *Z3* (version 4.8.5) [85] is used to produce the optimized solution through the proposed *SP&R* formulation. *SP&R* generates the "design layout" file with the information of FETs, nets (i.e., target nets for in-cell routing), and I/O pins (i.e., P_{EX}) from netlist of standard cell libraries. We choose 37 out of 69 cells from NanGate's 15*nm* open cell library [95] and 85 out of 183 cells from the ASAP7 library [3] to show the feasibility and the scalability of *SP&R* framework. The 15*nm* library is converted to the 7*nm* cell-equivalent architecture (6 horizontal routing tracks and 3 fins) of ASAP7 library for having the same number of routing tracks and fins. We tightly specify design parameters (MAR/EOL/VR/SHR/PRL/FST/T = 2/2/1.5/2/1/disable/1 or 2) for NanGate library to demonstrate innovative features of *SP&R* while the parameters of the ASAP7 library is specified to have the most similar routing result with the original library (MAR/EOL/VR/SHR/PRL/FST/T = 1/1/1.5/1/1/enable/1). The major DB of NanGate and ASAP7 libraries are SDB and DDB, respectively.

5.3.1 Experimental Results

Table 5.1 represents the scalability improvement stages of *SP&R* framework. Phase I refers to the base framework of our previous research [44]. Phase II includes a simple pre-processing based-on the BCP (Boolean constraint propagation), breaking design symmetry, and conditional assignment based on Phase I. Phase III and Phase IV perform localization of the routing region and objective partitioning based on Phase II and Phase III, respectively.

Trade-Off between Scalability and Key Metrics. The conditional localization (in Phase III) cuts the search space based on the assumption that the optimal routing path has the minimum metal length for each commodity. The objective partitioning (in Phase IV) separates the total

Table 5.1: Scalability improvement stages.

Stage	**Description**
Phase I	The framework of [44]
Phase II	Phase I + pre-processing + breaking design symmetry + conditional assignment
Phase III	Phase II + localization of the routing region
Phase IV	Phase III + objective priority in lexicographic order (Section 4.2.3)

Table 5.2: Experimental results presenting the trade-off between scalability and key metrics in *SP&R*: All values are on average. $T1/T2$ = tolerance 1/2 of the localization, #Var/#Con = the number of variables/constraints, inc./impr. = increment/improvement(in magnification) ratio (reference = Phase I), M_2 = the number of used $M2$ tracks.

	SMT Formulation		Key Metrics							Runtime	
	#Var	#Con	Size	Metal Length				M_2			
				Total	inc.	VIA	Metal	Avg.	inc.	Avg.(s)	impr.
Phase I	**14016.2**	**40000.7**	**6.91**	**117.89**	**0.00%**	**74.40**	**43.49**	**0.20**	**0%**	**75.14**	**1.0×**
Phase II	13439.5	38077.6								18.18	4.1×
Phase III ($T1$)	**13083.4**	**36856.7**		118.09	0.17%					7.03	10.7×
Phase III ($T2$)				**117.89**	**0.00%**					**8.94**	**8.4×**
Phase IV ($T1$)				118.11	0.19%		43.71			3.61	20.8×
Phase IV ($T2$)				**117.94**	**0.05%**	**74.29**	**43.66**			**3.80**	**19.8×**
Seq. P&R	**-**	**-**	**6.91**	**121.17**	**2.79%**	**75.77**	**45.40**	**0.31**	**55%**	**-**	**-**

metal length objective by a priority as described in Section 5.2. Therefore, these methods may affect the key metrics (i.e., cell size, #$M2$ tracks, and total metal length) of *SP&R*. Table 5.2 presents the experimental results showing the trade-off between scalability and key metrics of each improvement stage for the conventional sequential P&R simulation of 35 combinational logic cells in NanGate library [95]. Phase III and IV have split cases according to tolerance T of the conditional localization. The average runtime has improved up to 20.8× (75.14s@Phase I $\rightarrow$ 3.61s@Phase IV ($T1$)). In Phase III, the conditional localization with tolerance $T = 1$ ($T1$) results in 0.17% increment in the total metal length compared to Phase I. On the other hand, with tolerance $T = 2$ ($T2$), all key metrics are the same. This demonstrates that the conditional localization with $T = 2$ is not harmful to the key metrics. Though the objective partitioning in Phase IV results in the slightly-increased total metal length by 0.02% and 0.05% with $T1$ and $T2$, respectively, the length of VIA (which has a higher priority in separated objectives) is smaller than

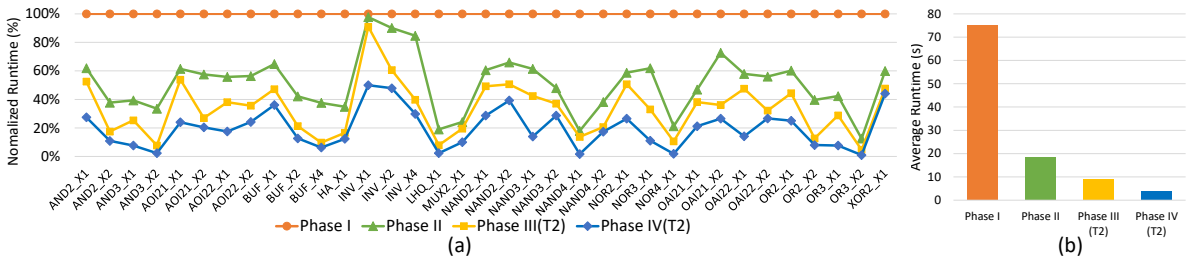


Figure 5.9: Contributions of each scalability improvement phase for runtime reduction. (a) Normalized runtime. (b) Average runtime.

or equal to Phase I. This indicates that the objective partitioning with a proper priority according to the weight of the original objective provides the equivalent optimization results to the original objective with a maximum gap of 0.05%. Despite the slight (up to 0.19%) degradation in the total metal length, all improvement stages present better results in terms of the total metal length and #$M2$ tracks compared to the results of the conventional sequential P&R approach.

Combinational Cells. Figure 5.9 visualizes the runtime improvement contributions on each improvement stage. The design symmetry breaking (Phase II) method brings significant improvement by cutting the search space for the most of combinational logic cells except the inverter-type cells (i.e., INV_X1, INV_X2, and INV_X4) as shown in Figure 5.9(a). This is because breaking design symmetry constraints is not applied to the cells with 1 FET in each P/N region. Meanwhile, the runtime to solve the problem depends on the random seed due to the heuristic aspects of SMT. Therefore, we select the best-achieved results as the runtime from the multiple random seeds for each cell. Figure 5.10 shows the variant of the runtime across 21 random seeds, depicted in box plots. For most of the cells, the runtime deviation tends to decrease by adding improvement stages. This demonstrates that the search space is effectively reduced with the scalability improvement constraints.

Large Sequential Cells. Figure 5.11(a) shows the key metrics of large sequential cells in NanGate library [95]. The number of FETs/nets in DFFSNQ_X1 and SDFFSNQ_X1 are 28/19 and 36/27, respectively. Due to the high complexity, the cell-partitioning constraints with six functional modules are applied to the improvement features of Phase IV. Since the

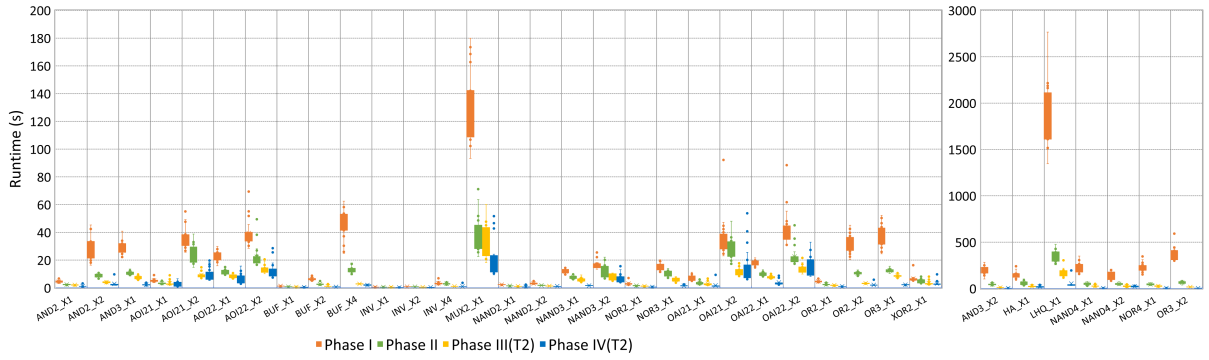
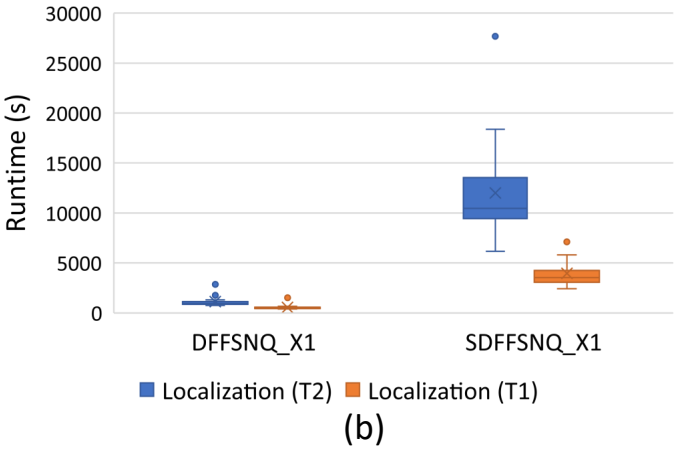


Figure 5.10: Statistical runtime visualization of combinational logic cells (21 random seeds).

(T : tolerance of localization)

	T	Key Metrics				
		Size	Metal Length			M_2
			Total	VIA	Metal	
DFFSNQ_X1	T1	23	612	340	272	3
	T2		606	336	270	
SDFFSNQ_X1	T1	29	702	388	314	
	T2	29	697	384	313	

(a)



(b)

Figure 5.11: Large sequential cells. (a) Key metrics statistics. (b) Runtime variation depicted in box plots (21 random seeds).

cell partitioning itself implies the breaking of design symmetry, the breaking design symmetry constraint is excluded. With the increased tolerance ($1 \rightarrow 2$) of the conditional localization and the objective partitioning, the total metal length as well as VIA length is decreased similarly to the Phase IV cases in Table 5.2. The decreased runtime deviation in Figure 5.11(b) with the decrease of the tolerance shows the reduction of the search space in the conditional localization. Examples of the cell partitioning based on the functionalities and the generated layout of SDFFSNQ_X1 cell are shown in Figure 5.12(a) and Figure 5.12(b), respectively. The cell is partitioned into and ordered with six functional modules as follows: DIN-MASTER-TRANS-SLAVE-CLOCK-DOUT.

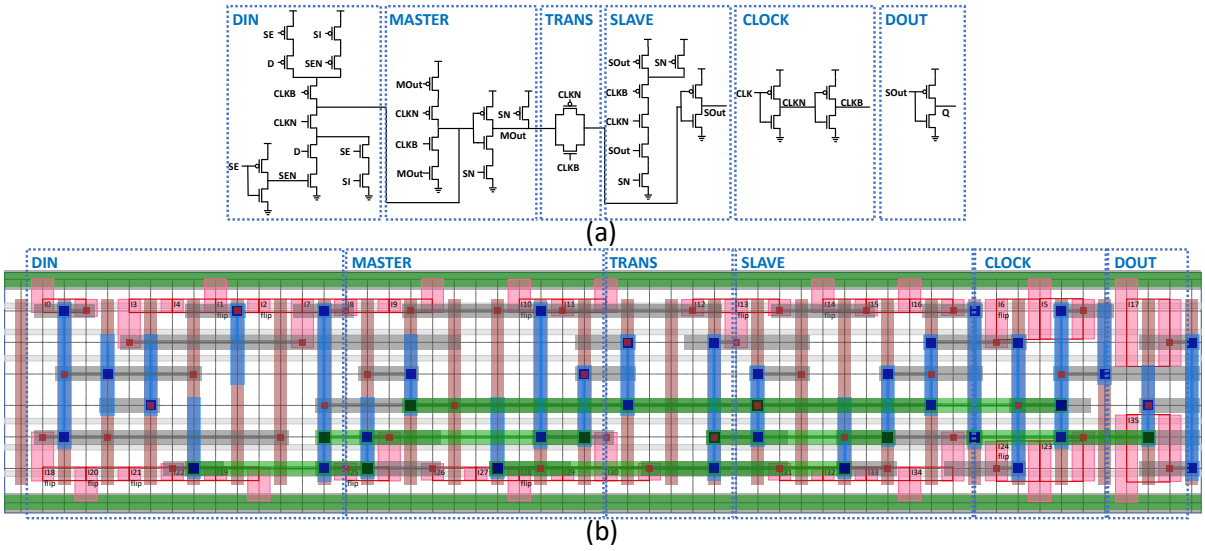


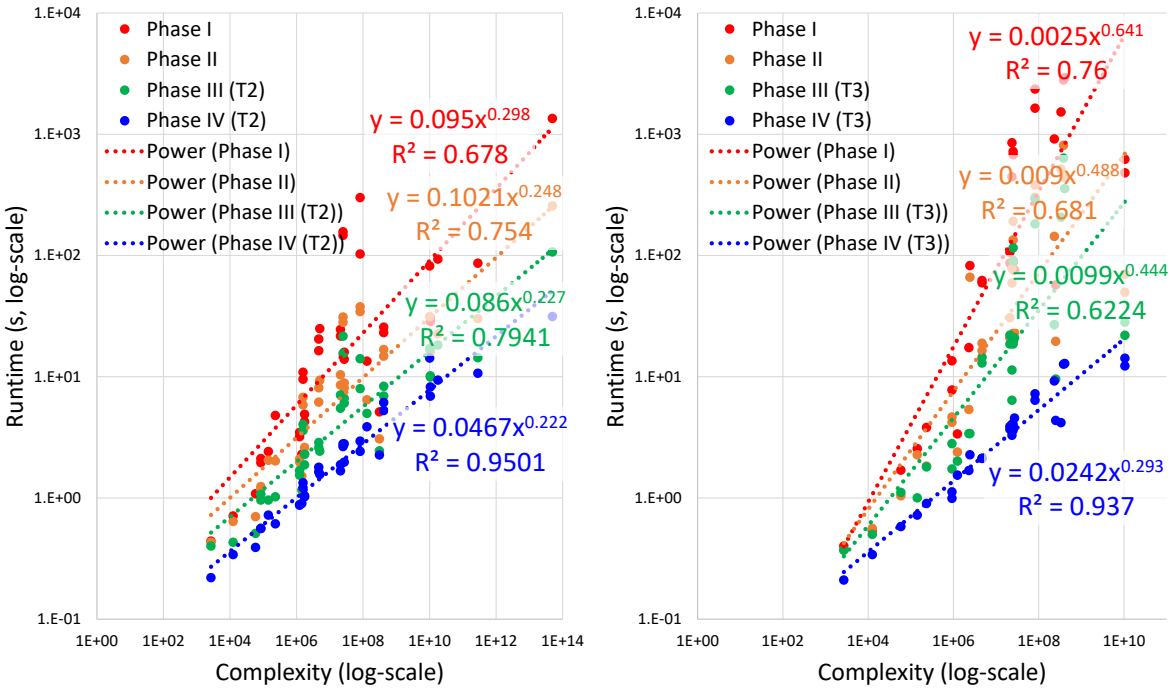
Figure 5.12: SDFFSNQ_X1. The largest cell in this work (29 CPPs). (a) Function module partitioning. (b) The generated layout (6168 seconds).

Scalability of SP&R Framework. The placement permutation is expressed in Expression (5.2). The number of clauses for routing is derived as Expression (5.3)[**?**]. In order to predict the runtime of SP&R, we test various structures combining Expressions (5.2) and (5.3) to maximize the correlation R^2. Using Expression (5.4), we achieve $R^2 = 0.9501$ for Phase IV on NanGate 35 combinational logic cells (Fig. 5.13(a)) and $R^2 = 0.937$ for Phase IV on ASAP7 on 30 combinational logic cells (Fig. 5.13(b)).

$$\textbf{Placement permutation :}\ O\Big((\frac{(X/2)!}{(X/2-N/2)!})^2\Big) \tag{5.2}$$

$$\textbf{Routing clauses :}\ O\big(X \cdot P\big) \tag{5.3}$$

$$\textbf{\textit{SP\&R} runtime prediction base :}\ O\Big(\frac{(X/2)!}{(X/2-N/2)!} \cdot X^2 \cdot P^2\Big) \tag{5.4}$$



(a) NanGate 35 combinational logic cells

(b) ASAP7 30 combinational logic cells

Figure 5.13: Scalability of *SP&R* framework for combinational cells (in log-scale).

5.3.2 Experimental Statistics of a Practical 7nm Cell Library

We show that the proposed *SP&R* satisfies both practicality and scalability through comparing key metrics and design features across the known layouts of 7*nm* ASAP7 standard cell library [3]. All results are generated with the proposed features of Phase IV ($T1$). For latch/flipflop cells, cell partitioning, crosstalk mitigation, and SDB in a crossover constraint are applied.

Combinational Logic Cells. Table 5.3 presents the results of 142 combinational logic cells in ASAP7 library. The #Cell refers to the number of variants of each cell type. The number of FETs in each cell ranges from 2 to 24 and the cell size is in the range of 3–30 CPPs. Compared to the known layouts of ASAP7 library, *SP&R* has similar or better results in terms of cell size and number of used $M2$ tracks. Two TIE cells are the two exceptions that require a gate cut structure (i.e., connecting different gate signals on the same gate column), which our framework

Table 5.3: *SP&R* results of 142 combinational logic cells from ASAP7 library without FET separation: #Cell = the number of variants of each celltype in column 1, #FET / #NET = the minimum / maximum number of FETs / Nets, M_2 = the number of used $M2$ tracks, Size/M_2/Runtime are on average value.

CellType	#Cell	#FET		#NET		Size		ASAP7[3]		SP&R		Runtime(s)	
		Min.	Max.	Min.	Max.	Min.	Max.	Size	M_2	Size	M_2	SP&R	[96]
INV/BUF	25	2	10	4	11	3	30	9.6	0.0	9.6	0.0	8.1	17.0
AND/OR	16	6	12	7	13	6	14	7.8	0.0	7.8	0.0	36.3	-
NAND/NOR	21	4	10	5	12	4	14	7.4	0.0	7.4	0.0	14.7	84.5
AOI/OAI	34	6	18	8	20	5	13	7.9	0.0	7.9	0.0	160.4	116.5
AO/OA	32	8	20	9	21	6	16	11.1	0.0	11.0	0.0	579.8	-
AOAI/OAOI	3	8	10	10	12	6	9	7.0	0.0	7.0	0.0	13.7	-
MAJ	3	10	12	10	11	7	10	8.7	0.0	8.7	0.0	14.2	-
XNOR/XOR	4	10	12	9	10	9	11	10.0	0.0	10.0	0.0	684.5	-
TIE	2	2	2	4	4	4	4	3.0	0.0	4.0	0.0	0.4	-
HA/FA	2	10	24	9	17	9	14	11.5	2.0	11.5	0.0	126.8	-
Total	**142**	**Average**						**8.84**	**0.03**	**8.83**	**0.00**	**203.4**	**-**

does not support. Though a fair comparison is not possible because of the differences in $M1$ routing (*SP&R*: uni-direction, ASAP7: bi-direction) and positions of gate contacts, cell size and the number of occupied $M2$ tracks can be compared to each other because the important routing resources such as the number of horizontal routing tracks and fins are the same and the design rule parameters in *SP&R* are carefully tuned to match the routability. Among 142 cells, 1 cell has reduced the number of $M2$ tracks by 4, and 2 cells have reduced cell size by 1 to 2 CPPs. Compared to the previous sequential approach [96], the average runtime of *SP&R* for the same cell types with equivalent complexity shows the reasonable overhead. The average runtime per cell is about 4 minutes.

Sequential Logic Cells. Table 5.4 presents the results of 23 sequential logic cells in ASAP7 library. For all sequential logic cells, *SP&R* obtains superior solutions that are smaller than or equal to the known layouts from ASAP7 library in terms of the cell size and the number of used $M2$ tracks. Fig. 5.3(b) displays a layout of DFFHQNx1 generated by *SP&R*. With the same cell size, the result of *SP&R* requires less $M2$ routing tracks than the known layout of Fig. 5.3(a). All sequential logic cells are generated within 42 minutes and the average runtime per cell is less

Table 5.4: *SP&R* results of 23 sequential logic cells from ASAP7 library : #Cell = the number of variants of each cell in column 1, #FET / #NET = the number of FETs / Nets, M_2 = the number of used *M*2 tracks.

CellType	#FET	#NET	ASAP7[3]		SP&R			[96]				
			Size	M2	Size	M2	Runtime	CellType	#FET	#NET	Size	Runtime
DHLx1	16	13	15	2	15	0	95.3	ELATN X1	12	11	12	1272
DHLx2	16	13	16	2	16	0	95.8	ELATS X1	10	11	10	1048
DHLx3	16	13	17	2	17	0	124.3	ELAT X1	12	11	12	1220
DLLx1	16	13	15	2	15	0	93.1	ELAT X3	12	11	16	2740
DLLx2	16	13	16	2	16	0	90.0	INV ELAT X1	14	12	16	3657
DLLx3	16	13	17	2	17	0	126.1	INV ELAT X3	14	12	20	654
DFFHQNx1	24	17	20	2	20	0	170.7	DFFQ X1	28	21	20	4351
DFFHQNx2	24	17	21	2	21	0	174.6	ESLATS X1	26	25	32	4217
DFFHQNx3	24	17	22	2	22	0	212.5	L1LATF X1	26	21	22	4155
DFFHQx4	26	18	25	2	25	0	342.5					
DFFLQNx1	24	17	20	2	20	0	173.2					
DFFLQNx2	24	17	21	2	21	0	197.5					
DFFLQNx3	24	17	22	2	22	0	210.8					
DFFLQx4	26	18	25	2	25	1	519.0					
SDFHx1	32	23	25	5	25	3	1880.4	ESLATN X1	32	25	36	6729
SDFHx2	32	23	26	4	26	3	2327.2	ESLAT X1	32	25	36	5763
SDFHx3	32	23	27	4	27	3	2466.5	ESLAT X3	32	25	36	4250
SDFHx4	32	23	31	3	28	3	2184.1	SDFFQS X1	32	27	24	31630
SDFLx1	32	23	25	4	25	2	1511.3					
SDFLx2	32	23	26	4	26	2	1479.5					
SDFLx3	32	23	27	4	27	2	1632.0					
SDFLx4	32	23	31	3	28	2	1824.8					
ASYNC_DFFHx1	32	23	26	6	26	2	2848.0					
average	**25.2**	**18.4**	**22.4**	**2.8**	**22.2**	**1.0**	**903.4**					

than 16 minutes. Compared to the previous work [96], *SP&R*'s cell generation shows the smaller runtime for the cells with the equivalent complexity.

5.4 Standard-cell Scaling Framework

In this section, we describe the detailed approaches of standard-cell scaling framework: (i) Framework overview; (ii) Sub-7*nm* cell architectures; (iii) Parametric conditional design rules; and (iv) Layout synthesis with guaranteed pin-accessibility.

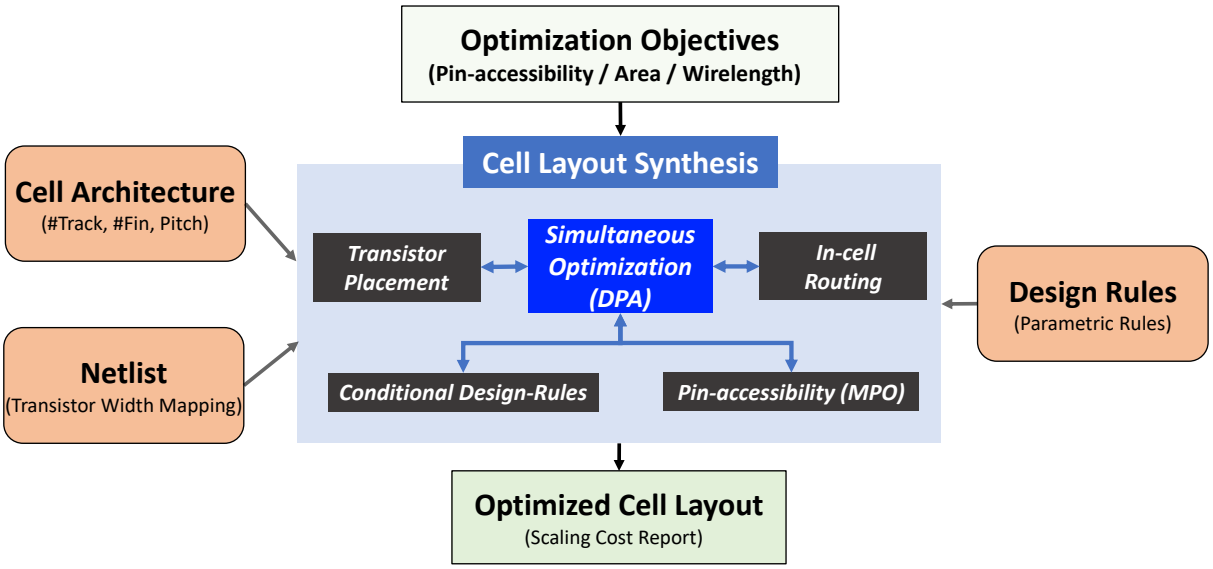


Figure 5.14: An overview of Standard-cell Scaling Framework.

5.4.1 Framework Overview

Figure 5.14 shows an overview of our scaling framework. The system contains three input files of 1) cell architectures: number of horizontal routing tracks, number of transistor fins, and the pitches of the tracks, 2) netlist: logic diagrams of library cells, and 3) design rules: conditional design rules are specified with the parameters of the rules. The cell layout synthesis operator reads these files and outputs the design with optimized cell area and wire length in lexicographic order with pin-accessibility constraints through a SMT-based state-of-the-art cell layout synthesis [44, 97]. To integrate various design constraints such as transistor placement, in-cell routing, conditional design rules, and pin-accessibility related constraint (i.e., MPO), we devise the DPA (dynamic pin allocation) scheme as a constraint without deploying any sequential/separate operations. By virtue of CSP (constraint satisfaction problem)'s exact solving, the standard cell synthesis framework generates an optimized cell layout within a single multi-objective optimization procedure through the SMT solving.

Table 5.5: Scaling Architecture Types. All numbers are in [*nm*].

Cell Architecture	CPP	MP	Cell Height
3 fins and 6 routing tracks (3F/6RT)	54	40	300
3 fins and 5 routing tracks (3F/5RT)			260
2 fins and 5 routing tracks (2F/5RT)	48	32	208
2 fins and 4 routing tracks (2F/4RT)			176
1 fin and 4 routing tracks (1F/4RT)	45	24	132
1 fin and 3 routing tracks (1F/3RT)			108

5.4.2 Sub-7*nm* Cell Architectures

Based on the layout configuration of Figure 4.3 (in the chapter 4), our standard cell scaling framework supports 6 kinds of cell architectures with different track number for standard-cell scaling scenarios in sub-7*nm* technology nodes as shown in Table 5.5. With reference to [98, 99, 2], we select the parametric cell architectures covering possible candidates in sub-7*nm* technology nodes. For Contact Poly Pitch (CPP), Metal Pitch (MP) and cell height information, we assume a hypothetical scaling pitch.

5.4.3 Parametric Conditional Design Rules

In this work, we cover representative conditional design rules for both EUV and multi-pattern technologies [5, 66, 44]. For placement, *Diffusion Sharing* and *Active-to-Active (AA)* rule are used to formulate the physical relation between the transistors. For routing, *Minimum Area Rule (MAR)*, *Via Rule (VR)* and *End-of-Line spacing (EOL)* are considered. For multi-pattern technologies, we include *Parallel Run Length* and *Step Height* rules for manufacturing SADP (Self-aligned double patterning) mask [40]. All conditional design rules are specified with the parameters of the rule. Figure 5.15 shows examples of parametric design rules. In our framework, all design rules are parameterized by the grid.

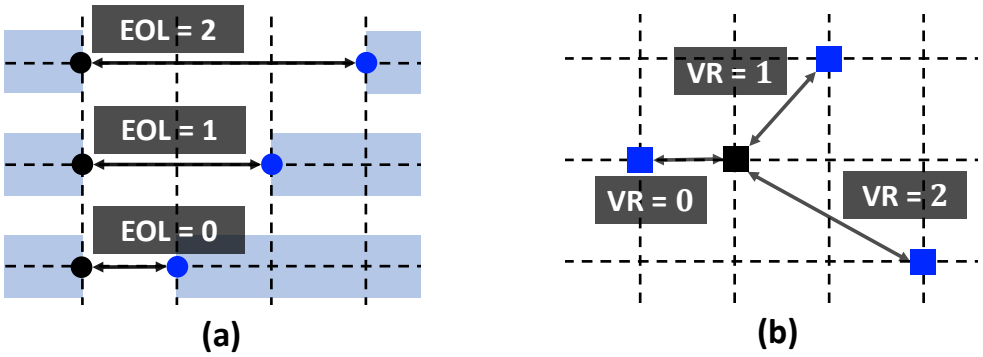


Figure 5.15: Examples of parametric design rules. (a) EOL. (b) VR. All number is in grid.

5.4.4 Layout Synthesis with Guaranteed Pin-accessibility

On top of the layout optimality of our framework through the simultaneous manner optimization [44], we devise boolean counter-based constraints to ensure the pin-accessibility, which are *Minimum I/O Pin Length (MPL)* and *Minimum I/O Pin Opening (MPO) rule*. With MPL and MPO, the cell layout has the guaranteed pin access positions irregardless of scaling parameters. Figure 5.16 show an example of MPL and MPO. To improve pin-accessibility in the example, the length of each pin is at-least-2 grids by MPL constraint, and each pin has at-least-2 accessible pin positions by MPO constraint.

Minimum I/O Pin Length (MPL). MPL rule defines the minimum number of metal segments of the commodity heading to the external pin P_{EX} on the $M1$ layer as shown in Figure 5.16(a). At-least 1 (**AL1**) metal segment on the $M1$ layer must be assigned to the commodity whose sink is P_{EX} as expressed in Constraint (5.5). Then, the metal segment on the $M1$ layer is extended to the minimum length defined by MAR. The vertices on the extended segments are the possible pin access points.

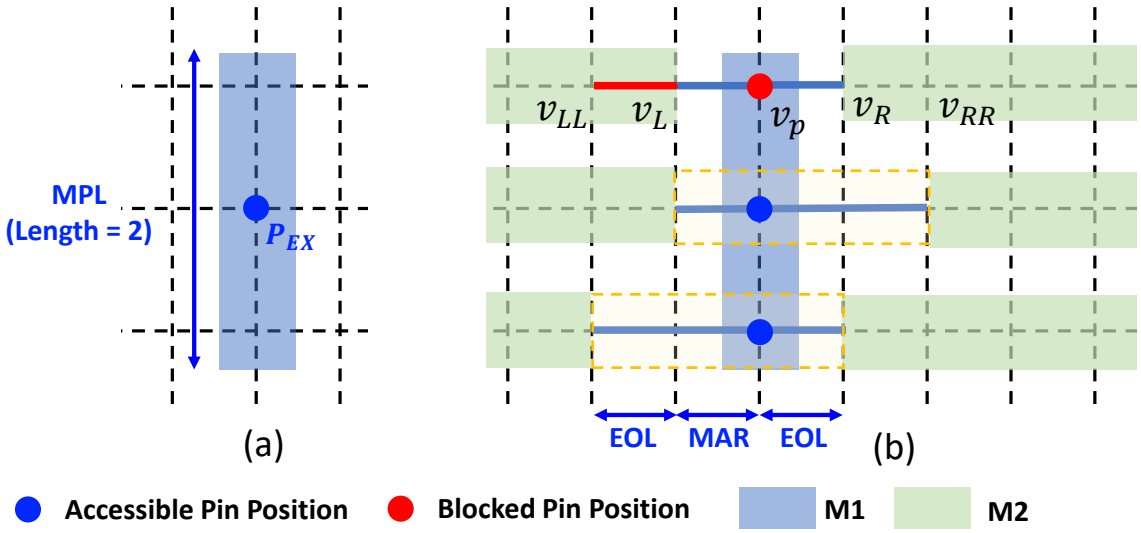


Figure 5.16: An example of conditional constraints for the pin-accessibility. (a) MPL rule with MAR=2. (b) MPO with EOL/MAR = 1/1.

$$\textbf{AL1}(m_{v,v_F}, m_{v,v_B}), \quad \text{if } f_m^n(v, v_D) = 1 \ , \ f_m^n(v, v_U) = 1$$
$$\forall v \in V_1, \ m = P_{EX} \tag{5.5}$$

Minimum I/O Pin Opening (MPO). MPO rule ensures the minimum number of unblocked access points (i.e., pin openings) from the $M2$ layer for each I/O pin. Figure 5.16(b) illustrates that each pin candidate v_p has to secure enough horizontal space on the $M2$ layer so that it can be accessed through the $M2$ layer without violating design rules such as the MAR and EOL. MPO considers each v_p as the possible pin opening if there is no routed metal segment in the opening mask (depicted in light yellow rectangles) on the $M2$ layer. MPO is a boolean cardinality constraint to ensure at-least-k (**ALk**) true pin opening indicator O_{v_p} among the possible candidates $Q(p)$ as described in Constraint (5.6). If there exist any edges $e_{v,u}^n$ on the $M2$ layer, MPO is not applied because the external pin p already has unblocked access points on the $M2$ layer.

$$\mathbf{ALk}\big(\{O_{v_p}|\ v_p \in Q(p)\}\big), \quad \text{if} \bigvee_{v\in V_2, u\in V_2} e_{v,u}^{n(p)} = 0 \tag{5.6}$$

For the example of Figure 5.16(b), the O_{v_p} is set as 1 (true) if there is no routed metal segment in the opening mask, whose length is the summation of MAR and EOL parameters (i.e., MAR + 2 $\times$ EOL), as shown in Constraint (5.7).

$$\overline{O_{v_p}} = \sum_{n\in N, n\neq n(p)} \Big(\big(e^n_{v_{LL},v_L} \vee e^n_{v_L,v_p} \vee e^n_{v_p,v_R}\big) \wedge \big(e^n_{v_L,v_p} \vee e^n_{v_p,v_R} \vee e^n_{v_R,v_{RR}}\big) \Big),$$
$$\forall v_p \in Q(p), \quad \begin{cases} v \in Q(p), & \text{if } e^n_{v_D,v_{DF}} = 1 \ , \ e^n_{v_D,v_{DB}} = 1 \\ v \notin Q(p), & \text{otherwise} \end{cases} , \forall v \in V_2 \tag{5.7}$$

5.5 Experiments for Standard-Cell Scaling Framework

In this section, we describe the experimental flows of our proposed framework: (i) Experimental setup and (ii) Experimental results.

5.5.1 Experimental Setup

The proposed framework is implemented in Perl/SMT-LIB 2.0 standard-based formula and validated on a Linux workstation with 2.4GHz Intel Xeon E5-2620 CPU and 256GB memory. The single-threaded SMT Solver *Z3* (version 4.8.5) is used to produce the optimized cell layout for each case.

Transistor Width Mapping. A proper transistor width mapping roadmap is required for the practical comparison since we utilize the netlist information of 15*nm* cell architecture [95] with 7 fins in our experiments. Table 5.6 shows a hypothetical but realistic mapping roadmap for

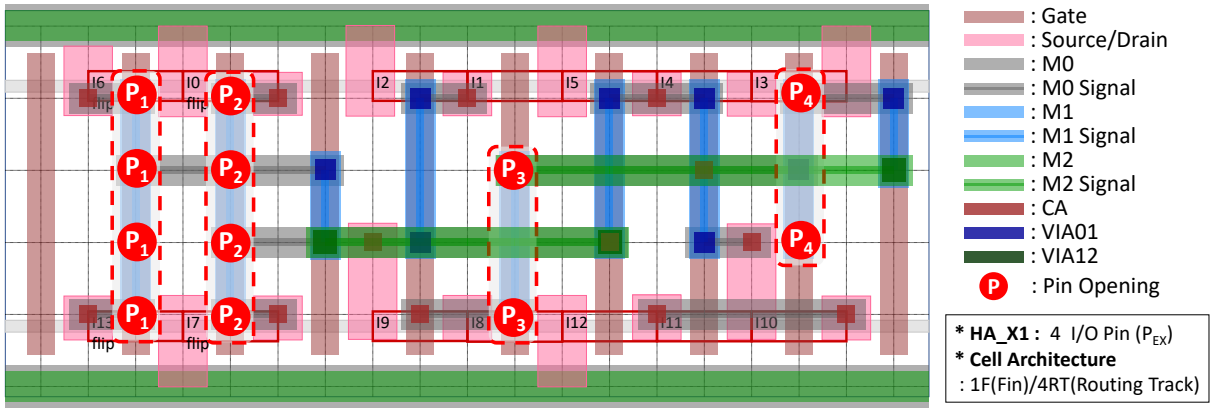


Figure 5.17: An example of cell layout with guaranteed pin-accessibility: HA_X1.

transistor width sizing.

Table 5.6: Hypothetical Transistor Width Mapping.

Architecture Fin Type			
7 fins [95]	3 fins	2 fins	1 fin
1/2/3	1		1
4/5	2	2	
6/7	3		

Reference Conditional Design Rules. To achieve the practicality of the framework in the experiments, we refer to the EUV-compatible conditional design rules [100] excluding the multi-pattern technology-related design rules. Note that the experiment parameters of conditional design rules are as follows: AA = 1, MAR = 1, EOL = 1, and VR = 1.

5.5.2 Experimental Results

We validate our framework to check the guaranteed pin-accessibility and explore the scaling effect of standard-cell from two perspectives (i.e., the track and design-rule knob).

Optimized Layout with Guaranteed Pin-accessibility. Figure 5.17 shows an example of scaled cell layout generated by our framework. All pins are successfully satisfying MPL

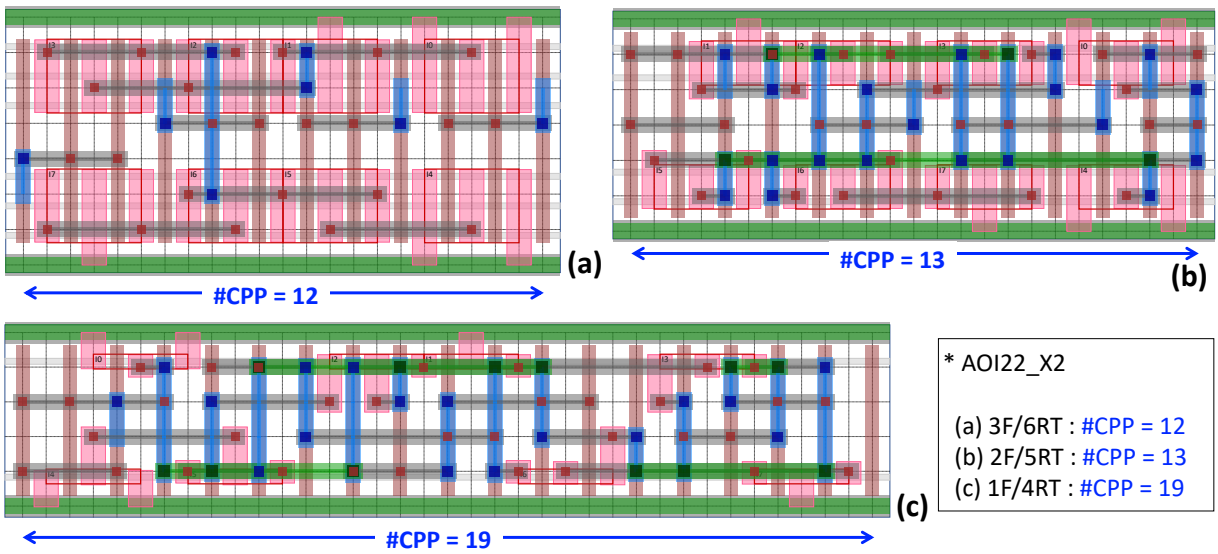


Figure 5.18: An example of standard-cell layout with scaled track number: AOI22_X2. (a) #CPP = 12 with 3F/6RT. (b) #CPP = 13 with 2F/5RT. (c) #CPP = 19 with 1F/4RT. We assume that CPP and MP are fixed.

(minimum I/O pin length) and MPO (minimum I/O pin opening) constraint, resulting in 4,4,2[1] and 2 pin openings for P_1, P_2, P_3 and P_4, respectively.

Process Scaling Effect (The track knob). We explore the process scaling effect by utilizing cell architectures of Table 5.5 and transistor width mapping of Table 5.6. Without pitch scaling (i.e. CPP and MP are fixed), the scaled cell layout typically requires more CPPs due to the lack of resource for in-cell routing when the number of track is reduced. Figure 5.18 shows an example using AOI22_X2. In Figure 5.18(a), AOI22_X2 cell has 12 #CPP (Contact Poly Pitch) with 3F/6RT architecture. The #CPP increases from 12 to 13 (Figure 5.18(b) : 2F/5RT) and then to 19 (Figure 5.18(c) : 1F/4RT) as the cell architecture is scaled down. Therefore, the reduction on routing tracks is offset by the increase in #CPP.

Figure 5.19 shows overall process scaling effect on the basis of libraries consisting of 32 cells. The blue bars of Figure 5.19 represent the area benefits without pitch scaling for each cell architecture. Without pitch scaling and DTCO, the area benefit is limited by 13%

[1]In the example, the routed elements on $M2$ track for P_3 are also the possible pin access openings. In that case, MPO is satisfied by extra openings.

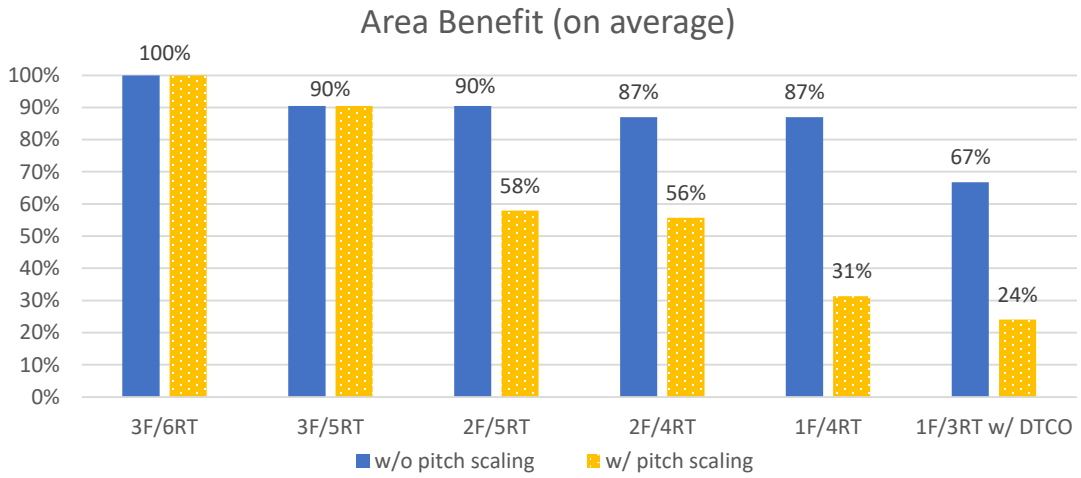


Figure 5.19: Area benefit through the track scaling.

on average although the track number is reduced by 2 (3F/6RT $\rightarrow$ 1F/4RT). the yellow bars of Figure 5.19 represent the area benefits with pitch scaling (CPP/MP of Table 5.5) for each cell architecture. With pitch scaling, the area benefit can be achieved up to 69% on average in the 1F/4RT architecture. Since our framework diagnoses that the most of cell libraries are not feasible (unroutable) in the 1F/3RT architecture with the reference design rule setting (AA=1/MAR=1/EOL=1/VR=1), we relax the design rules (EOL=1/VR=1 $\rightarrow$ EOL=0/VR=0) to get the area benefit (76%) for 1F/3RT. With DTCO and pitch scaling, up to 76% on average of area benefit can be achieved in the 1F/3RT architecture.

Figure 5.20 enumerates the area comparison between standard-cell architectures. In 2F/4RT architecture, some of 2F/4RT-based layouts (e.g., NAND4_X2, AOI22_X2, or OAI22_X2) is larger than 2F/5RT-based layouts although the area on average of 2F/4RT is less than that of 2F/5RT. For these cases, the designer should consider DTCO to get the further area shrink since the pitch scaling strategy is no more efficient way in the area reduction.

DTCO (The design-rule knob). To achieve additional area shrink, DTCO (Design technology co-optimization) such as design rule relaxation is essential in standard-cell scaling. As our framework provide the parametric modification of design rules, we explore DTCO effect by tuning the conditional design rules.

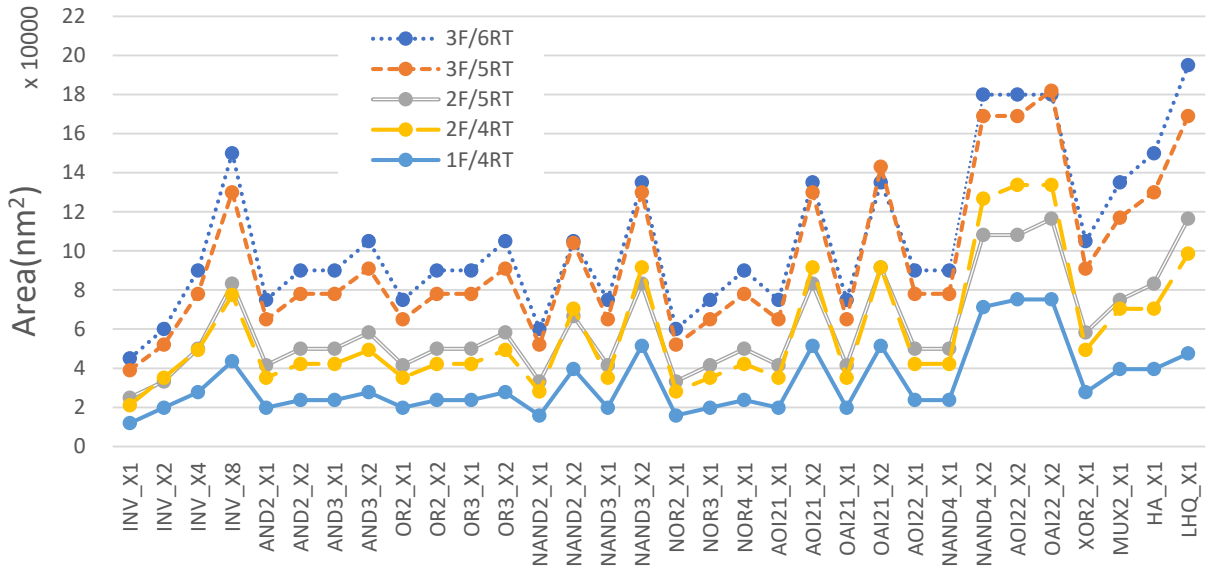


Figure 5.20: Area comparison between cell architectures with fixed design rules.

Figure 5.21 shows an example of DTCO using standard cell OAI22_X2. In Figure 5.21(a), we have the synthesized layout with the cell architecture of 3F/6RT. The number of columns (#CPP: Contact Poly Pitch) is 12. In Figure 5.21(b), we change the cell architecture to 2F/4RT. #CPP increases to 19 in order to keep the pin-accessibility with given design rules. The area scaling can only be obtained by the pitch shrinkage to achieve 26% reduction. In Figure 5.21(c), we relax the design rules (EOL=1/VR=1 $\rightarrow$ EOL=0/VR=0) and reduce #CPP back to 12. Together with pitch shrinkage, we achieve 53% area reduction. The process allows us to negotiate between the technology rules and the physical design for the area reduction.

For the cases of 2F/4RT architecture, we explore the DTCO effect as shown in Figure 5.22. Figure 5.22(a) shows the extra area benefit through the design rule relaxation for each cell design. While 44% on average of area reduction is possible through pitch scaling only, total 52% on average of area reduction can be achieved by applying both the DTCO and the pitch scaling as shown in Figure 5.22(b).

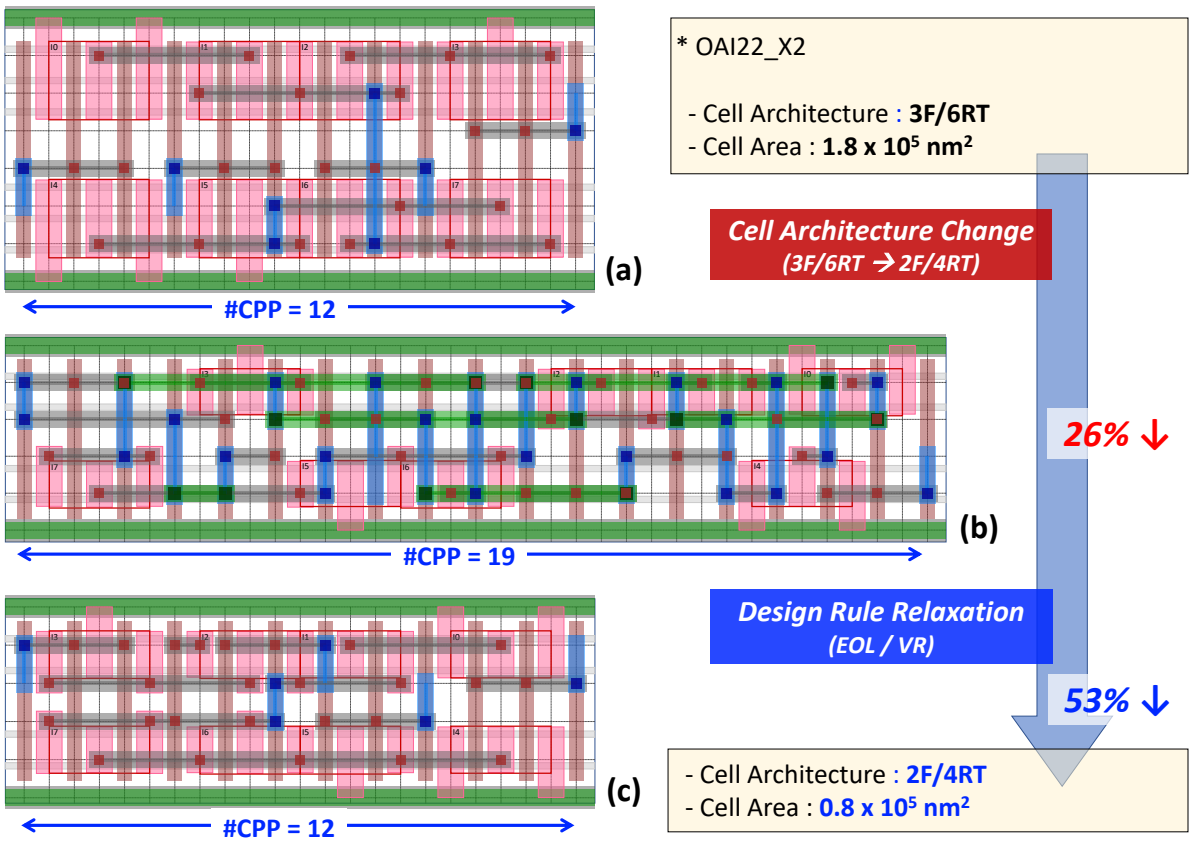


Figure 5.21: An example of DTCO: OAI22_X2. (a) Reference architecture. (b) Cell architecture change(3F/6RT → 2F/4RT). (c) Design rule relaxation.

5.6 Conclusion

In this chapter, we describe our orchestrated tactics to improve both practicality and scalability of our standard cell synthesis framework. We develop practical cell-design features to further optimize cell size and to guarantee the stable operations of timing-critical cells. We improve the scalability of our framework by introducing several search-space reduction techniques exploiting the nature of standard cell design, resulting in the generation of a whole standard cell library. We show that our framework successfully produces DRC-clean optimal layouts with substantial design features. *SP&R* achieves an average of 20.8× runtime improvement over the previous work [44] by exchanging less than 0.2% of the total metal length. We demonstrate that our framework successfully accomplishes a wide variety of cell-layout designs, up to 29 CPPs,

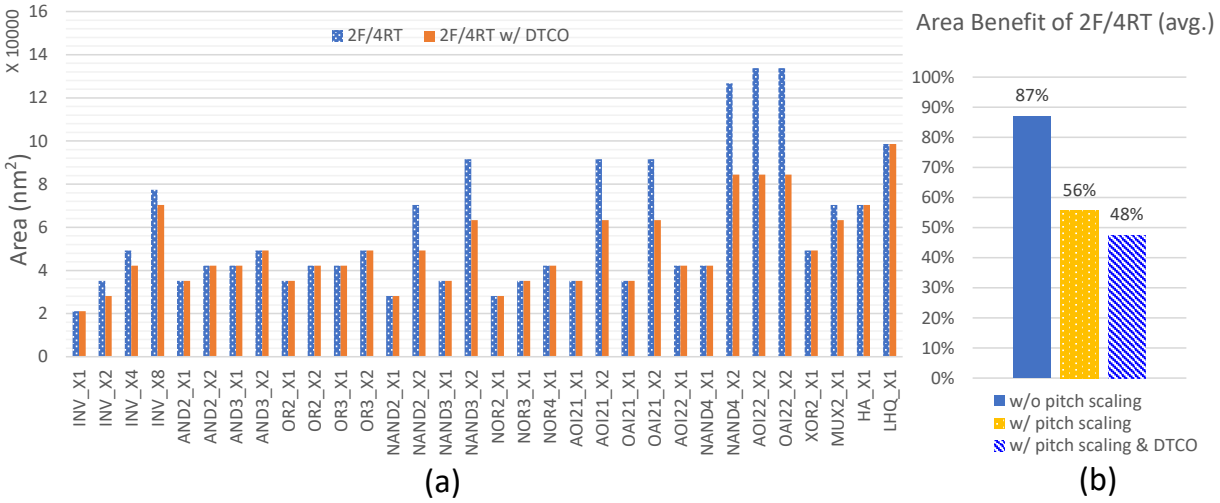


Figure 5.22: Area benefit from DTCO through the design rule relaxation (EOL/VR). (a) Statistics for each standard cell. (b) Area benefit on average by utilizing 2F/4RT architecture and DTCO.

36 FETs, 27 nets, and 92 commodities, within 1.75 hours for the largest cell (SDFFSNQ_X1, Figure 5.12).

Also, we propose a new scaling framework of standard-cell layout with guaranteed pin-accessibility providing the optimized cell design through a simultaneous-P&R methodology. From the pin-accessibility guarantees, i.e., MPL and MPO constraint, and the practical considerations (e.g., diverse cell architectures, realistic transistor width mapping and conditional design rule), our framework can explore the entire scaling effect of standard-cell design across sub-7*nm* technology nodes. The scaled cell layout can offer an early ”go/no-go” decision opportunity for the process migration procedures. In the next chapter, we summarize our contributions and discuss the future work.

This chapter contains materials from following publications: “Standard-Cell Scaling Framework with Guaranteed Pin-Accessibility”, by Chung-Kuan Cheng, Daeyeal Lee, and Dongwon Park, which appears in IEEE International Symposium on Circuits and Systems, October 2020. “SP&R: SMT-based Simultaneous Place-&-Route for Standard Cell Synthesis of Advanced Nodes”, by Daeyeal Lee, Dongwon Park, Chiatung Ho, Ilgweon Kang, Hayoung

Kim, Sicun Gao, Bill Lin, and Chung-Kuan Cheng, which appears in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, November 2020. The dissertation author was the primary investigator and author of these papers.

Chapter 6

Summary and Future Work

6.1 Thesis Summary

This thesis describes new design methodologies and its automated frameworks in three topics: (1) routability analysis and diagnosis framework in detailed routing (inter-cell place-and-route), (2) standard cell synthesis framework (in-cell place-and-route), and (3) parametric standard cell scaling study in terms of DTCO.

Chapter 2 and Chapter 3 present a new design methodology for fast turnaround in analyzing the routing feasibility of the given layout (Analysis) and a novel methodology to provide comprehensive diagnosis information of DRVs (Diagnosis). Our frameworks efficiently identify the design rule-correct routability by solving the proposed ILP and SAT formulation. We develop a new SAT-friendly ILP-based detailed routing formulation satisfying conditional design rules. During our ILP-to-SAT conversion, we reduce the complexity of the SAT problem by utilizing SAT encoding techniques, logic minimizer, and preprocessing. We demonstrate that our SAT-based routability analysis and diagnosis frameworks produce precise assessment/diagnosis of the design rule-correct routability, within 0.02% of ILP runtime on average. We show that our frameworks handle up to $210 \times 211 \times 758$ and $180 \times 151 \times 518$ case for routability analysis and

diagnosis, respectively.

Chapter 4 and Chapter 5 present a new SMT-based standard cell-layout design framework that satisfies both practicality and scalability. Our framework provides fully automated procedures generating the optimal cell layouts, after exploring the place-and-route combined search space based on the proposed DPA scheme. We develop practical cell-design features to further optimize cell size and to guarantee the stable operations of timing-critical cells. We improve the scalability of our framework by introducing several search-space reduction techniques exploiting the nature of standard cell design, resulting in the generation of a whole standard cell library. We show that our framework successfully produces DRC-clean optimal layouts with substantial design features. *SP&R* achieves an average of 20.8× runtime improvement over the previous work [44] by exchanging less than 0.2% of the total metal length. We demonstrate that our framework successfully accomplishes a wide variety of cell-layout designs, up to 29 CPPs, 36 FETs, 27 nets, and 92 commodities, within 1.75 hours for the largest cell (SDFFSNQ_X1, Figure 5.12).

In particular, Chapter 5 presents a new scaling framework of standard-cell layout with guaranteed pin-accessibility providing the optimized cell design through a simultaneous-P&R methodology. From the pin-accessibility guarantees and the practical considerations (e.g., diverse cell architectures, realistic transistor width mapping and conditional design rule), our framework can explore the entire scaling effect of standard-cell design across sub-7*nm* technology nodes. The scaled cell layout can offer an early ”go/no-go” decision opportunity for the process migration procedures.

6.2 Future Directions

As the future work, we have the plan to strengthen our optimization objectives, in terms of routability, of standard cell synthesis framework for extremely scaled technology nodes (e.g., Complementary-FET cell architecture [101, 102]), because the routability for both in-cell routing

and inter-cell routing in DR is one of most critical bottlenecks across all physical design stages for beyond 7*nm* technology nodes due to the extremely limited routing resources (including pin access resources). Furthermore, our frameworks can be extended to combine two key stages of physical design, which are detailed-placement and detailed-routing. With integrated constraints for detailed-placement and detailed-routing together, we can propose better DRV refined solutions in terms of both the feasibility and the optimality for target design metrics. As far as we know, there is no such methodology with concurrent detailed-placement and detailed-routing considerations. With this in mind, the rest of this section provides the details of our future directions.

6.2.1 Routability-driven standard cell synthesis in extremely scaled technology nodes

For extremely scaled technology nodes such as 5*nm* or 3*nm*, breakthrough DTCO methodologies are required because scaling methodologies for conventional cell structure are already exhausted for feasible standard cell layout. As depicted in Figure 1.2, CFET (Complimentart-FET) can be promising cell architecture replacing conventional FET architecture (so-called system technology co-optimization (STCO)). For the standard cell synthesis automation for CFET, we can extend our DPA (dynamic pin allocation) scheme by considering the unique stackable-FET structure of CFET. To mitigate extremely limited routability (including pin-accessibility) comes from the intrinsic stacked FET structure, routability-driven optimization features are essential to achieve feasible cell layout in entire physical design stages. This work has been published in IEEE/ACM 2020 International Conference on Computer-Aided Design [103], but this dissertation does not cover the detailed contents.

6.2.2 Concurrent refinement through simultaneous detailed-placement and detailed-routing

we can propose a concurrent optimization framework with respect to detailed-placement (DP), detailed-routing (DR), and standard cell synthesis together to reduce turnaround time of DR stage. Our multi-objective optimization synthetically considers cell adjustment minimization (for DP), metal length minimization (for DR), and pin-shape modification of standard cell (for standard cell library). By virtue of exhaustive exploration for integrated CSPs with respect to constraints of DP, DR, and conditional design rules, our SMT-based optimization framework resolve each DRV efficiently, resulting in an optimized place-and-route layout in terms of perturbation-minimizing multiple objectives. To the best of our knowledge, our framework is the first suggestion considering simultaneously DP, DR stage, and standard cell synthesis together without deploying any sequential or separate operations between main stages of physical design.

Bibliography

[1] A. Olofsson, "Silicon compilers-version 2.0," *keynote, Proc. ISPD*, 2018.

[2] S. M. Y. Sherazi, M. Cupak, P. Weckx, O. Zografos, D. Jang, P. Debacker, D. Verkest, A. Mocuta, R. H. Kim, A. Spessot, and J. Ryckaert, "Standard-cell design architecture options below 5nm node: The ultimate scaling of FinFET and Nanosheet," in *Design-Process-Technology Co-optimization for Manufacturability XIII*, J. P. Cain, Ed., vol. 10962, International Society for Optics and Photonics. SPIE, 2019, pp. 1 – 15.

[3] V. Vashishtha, M. Vangala, and L. T. Clark, "ASAP7 predictive design kit development and cell design technology co-optimization," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 992–998.

[4] Cadence Innovus User Guide, http:/www.cadence.com.

[5] I. Kang, D. Park, C. Han, and C.-K. Cheng, "Fast and precise routability analysis with conditional design rules," in *Proceedings of the 20th System Level Interconnect Prediction Workshop*, 2018, pp. 1–8.

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

[7] Gurobi Optimizer, https://www.gurobi.com/products/gurobi-optimizer/.

[8] P. Cremer, S. Hougardy, J. Schneider, and J. Silvanus, "Automatic cell layout in the 7nm era," in *Proceedings of the 2017 ACM on International Symposium on Physical Design*, 2017, pp. 99–106.

[9] K. Han, A. B. Kahng, and H. Lee, "Evaluation of beol design rule impacts using an optimal ilp-based detailed router," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2015, pp. 1–6.

[10] X. Jia, Y. Cai, Q. Zhou, and B. Yu, "A multicommodity flow-based detailed router with efficient acceleration techniques," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 217–230, 2017.

[11] K. Salimifard and S. Bigharaz, "The multicommodity network flow problem: state of the art classification, applications, and solution methods," *Operational Research*, pp. 1–47, 2020.

[12] I. Kang, "Floorplan representation, global placement, and routability analysis for vlsi layout design automation," Ph.D. dissertation, UC San Diego, 2018.

[13] N. Ryzhenko, S. Burns, A. Sorokin, and M. Talalay, "Pin access-driven design rule clean and dfm optimized routing of standard cells under boolean constraints," in *Proceedings of the 2019 International Symposium on Physical Design*, 2019, pp. 41–47.

[14] L. Luo and M. D. Wong, "Ordered escape routing based on boolean satisfiability," in *2008 Asia and South Pacific Design Automation Conference*. IEEE, 2008, pp. 244–249.

[15] H. Fraisse, A. Joshi, D. Gaitonde, and A. Kaviani, "Boolean satisfiability-based routing and its application to xilinx ultrascale clock network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2016, pp. 74–79.

[16] S. Mukherjee and S. Roy, "Sat based multi pin net detailed routing for fpga," in *2010 International Symposium on Electronic System Design*. IEEE, 2010, pp. 141–146.

[17] 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," in *International Workshop on System Level Interconnect Prediction*. IEEE, 2011, pp. 1–7.

[18] V.-H. Nguyen, "SAT Encodings of Finite CSPs," 2015.

[19] A. M. Frisch and P. A. Giannaros, "SAT encodings of the at-most-k constraint. some old, some new, some fast, some slow," in *Proc. of the tenth int. workshop of constraint modelling and reformulation*, 2010, p. 36.

[20] W. Klieber and G. Kwon, "Efficient cnf encoding for selecting 1 from n objects," in *Proc. International Workshop on Constraints in Formal Verification*, 2007.

[21] S. Darras, G. Dequen, L. Devendeville, B. Mazure, R. Ostrowski, and L. Sais, "Using boolean constraint propagation for sub-clauses deduction," in *International Conference on Principles and Practice of Constraint Programming*. Springer, 2005, pp. 757–761.

[22] M. H. Liffiton and K. A. Sakallah, "Algorithms for computing minimal unsatisfiable subsets of constraints," *Journal of Automated Reasoning*, vol. 40, no. 1, pp. 1–33, 2008.

[23] J. Marques-Silva, "Computing minimally unsatisfiable subformulas: State of the art and future directions." *Journal of Multiple-Valued Logic & Soft Computing*, vol. 19, 2012.

[24] C. Barrett and C. Tinelli, "Satisfiability modulo theories," in *Handbook of Model Checking*. Springer, 2018, pp. 305–343.

[25] A. Grimmer and R. Wille, "Designing application-specific architectures," in *Designing Droplet Microfluidic Networks*. Springer, 2020, pp. 83–98.

[26] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "Smc: Satisfiability modulo convex programming," *Proceedings of the IEEE*, vol. 106, no. 9, pp. 1655–1679, 2018.

[27] A. Cimatti, L. Geatti, A. Griggio, G. Kimberly, and S. Tonetta, "Safe decomposition of startup requirements: Verification and synthesis," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2020, pp. 155–172.

[28] L. W. Rizkallah, M. F. Ahmed, and N. M. Darwish, "Smt-lh: A new satisfiability modulo theory-based technique for solving vehicle routing problem with time window constraints," *The Computer Journal*, vol. 63, no. 1, pp. 91–104, 2020.

[29] S. M. Saif, M. Dessouky, M. W. El-Kharashi, H. Abbas, and S. Nassar, "Pareto front analog layout placement using satisfiability modulo theories," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 1411–1416.

[30] S. Banerjee, A. Ratna, and S. Roy, "Satisfiability modulo theory based methodology for floorplanning in vlsi circuits," in *2016 Sixth International Symposium on Embedded Computing and System Design (ISED)*. IEEE, 2016, pp. 91–95.

[31] B. Nikolaj and D. P. Anh, "vz-an optimizing smt solver," in *Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, 2015, pp. 194–199.

[32] R. Sebastiani and P. Trentin, "Optimathsat: A tool for optimization modulo theories," *Journal of Automated Reasoning*, pp. 1–38, 2018.

[33] J. Crawford, M. Ginsberg, E. Luks, and A. Roy, "Symmetry-breaking predicates for search problems," *KR*, vol. 96, pp. 148–159, 1996.

[34] D. Déharbe, P. Fontaine, S. Merz, and B. W. Paleo, "Exploiting symmetry in smt problems," in *International Conference on Automated Deduction*. Springer, 2011, pp. 222–236.

[35] J. P. Galeotti, N. Rosner, C. G. L. Pombo, and M. F. Frias, "Taco: Efficient sat-based bounded verification using symmetry breaking and tight bounds," *IEEE Transactions on Software Engineering*, vol. 39, no. 9, pp. 1283–1307, 2013.

[36] Y. Ding, C. Chu, and W.-K. Mak, "Pin accessibility-driven detailed placement refinement," in *Proceedings of the 2017 ACM on International Symposium on Physical Design*, 2017, pp. 133–140.

[37] J. Seo, J. Jung, S. Kim, and Y. Shin, "Pin accessibility-driven cell layout redesign and placement optimization," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

[38] Y. A. Badr, K.-w. Ma, and P. Gupta, "Layout pattern-driven design rule evaluation," *Journal of Micro/Nanolithography, MEMS, and MOEMS*, vol. 13, no. 4, p. 043018, 2014.

[39] 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," in *Design for Manufacturability through Design-Process Integration VI*, vol. 8327. International Society for Optics and Photonics, 2012, p. 832703.

[40] Y. Ma, J. Sweis, H. Yoshida, Y. Wang, J. Kye, and H. J. Levinson, "Self-aligned double patterning (sadp) compliant design flow," in *Design for Manufacturability through Design-Process Integration VI*, vol. 8327. International Society for Optics and Photonics, 2012, p. 832706.

[41] ITRS Report 2015, http://www.itrs2.net/itrs-reports.html.

[42] 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 (TODAES)*, vol. 21, no. 3, pp. 1–21, 2016.

[43] Cadence Design Systems, Inc., Sr Software Engineering Group Director, *personal communication*, 2019.

[44] D. Park, D. Lee, I. Kang, S. Gao, B. Lin, and C.-K. Cheng, "SP&R: Simultaneous Placement and Routing framework for standard cell synthesis in sub-7nm," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2020, pp. 345–350.

[45] D. Shi and A. Davoodi, "Improving detailed routability and pin access with 3d monolithic standard cells," in *Proceedings of the 2017 ACM on International Symposium on Physical Design*, 2017, pp. 107–112.

[46] V. Yutsis, I. S. Bustany, D. Chinnery, J. R. Shinnerl, and W.-H. Liu, "Ispd 2014 benchmarks with sub-45nm technology rules for detailed-routing-driven placement," in *Proceedings of the 2014 on International symposium on physical design*, 2014, pp. 161–168.

[47] 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," in *Proceedings of the 2015 Symposium on International Symposium on Physical Design*, 2015, pp. 157–164.

[48] T. Taghavi, Z. Li, C. Alpert, G.-J. Nam, A. Huber, and S. Ramji, "New placement prediction and mitigation techniques for local routing congestion," in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2010, pp. 621–624.

[49] 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," in *Proceedings of the 2015 Symposium on International Symposium on Physical Design*, 2015, pp. 149–156.

[50] 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," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

[51] M. M. Ozdal, "Detailed-routing algorithms for dense pin clusters in integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 3, pp. 340–349, 2009.

[52] F. A. Aloul, A. Ramani, I. L. Markov, and K. A. Sakallah, "Generic ilp versus specialized 0-1 ilp: An update," in *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, 2002, pp. 450–457.

[53] R. Carden, J. Li, and C.-K. Cheng, "A global router with a theoretical bound on the optimal solution," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 2, pp. 208–216, 1996.

[54] 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*, vol. 34, no. 5, pp. 699–712, 2015.

[55] M. N. Velev, "Exploiting hierarchy and structure to efficiently solve graph coloring as sat," in *2007 IEEE/ACM International Conference on Computer-Aided Design*. IEEE, 2007, pp. 135–142.

[56] G.-J. Nam, F. Aloul, K. A. Sakallah, and R. A. Rutenbar, "A comparative study of two boolean formulations of fpga detailed routing constraints," *IEEE Transactions on Computers*, vol. 53, no. 6, pp. 688–696, 2004.

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

[58] LEF/DEF Language Reference, http://www.ispd.cc/contests/18/lefdefref.pdf.

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

[60] G. Audemard and L. Simon, "Glucose and syrup in the sat race 2015," *SAT Race*, 2015.

[61] M. J. Heule, M. J. Järvisalo, and M. Suda, "Proceedings of sat competition 2018," 2018.

[62] W.-H. Liu, S. Mantik, W.-K. Chow, Y. Ding, A. Farshidi, and G. Posser, "Ispd 2019 initial detailed routing contest and benchmark with advanced routing rules," in *Proceedings of the 2019 International Symposium on Physical Design*, 2019, pp. 147–151.

[63] J. Zhang, T. Li, and S. Li, "Application and analysis of unsatisfiable cores on circuits synthesis," in *2015 Seventh International Conference on Advanced Computational Intelligence (ICACI)*. IEEE, 2015, pp. 407–410.

[64] D. Kroening and O. Strichman, *Decision Procedures: An Algorithmic Point of View*. Springer, 2016.

[65] P. Zarkesh-Ha, J. A. Davis, W. Loh, and J. D. Meindl, "Prediction of interconnect fan-out distribution using rent's rule," in *Proceedings of the 2000 international workshop on System-level interconnect prediction*, 2000, pp. 107–112.

[66] D. Park, I. Kang, Y. Kim, S. Gao, B. Lin, and C.-K. Cheng, "Road: Routability analysis and diagnosis framework based on sat techniques," in *Proceedings of the 2019 International Symposium on Physical Design*, 2019, pp. 65–72.

[67] O. Bastert and C. Matuszewski, "Layered drawings of digraphs," in *Drawing graphs*. Springer, 2001, pp. 87–120.

[68] A. Belov and J. Marques-Silva, "Muser2: An efficient mus extractor," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 8, no. 3-4, pp. 123–128, 2012.

[69] N. Eén and A. Biere, "Effective preprocessing in sat through variable and clause elimination," in *International conference on theory and applications of satisfiability testing*. Springer, 2005, pp. 61–75.

[70] A. P. Jacob, R. Xie, M. G. Sung, L. Liebmann, R. T. Lee, and B. Taylor, "Scaling challenges for advanced cmos devices," *International Journal of High Speed Electronics and Systems*, vol. 26, no. 01n02, p. 1740001, 2017.

[71] C.-K. Cheng, D. Lee, and D. Park, "Standard-Cell Scaling Framework with Guaranteed Pin-Accessibility," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020, pp. 1–5.

[72] A. B. Kahng, H. Lee, and J. Li, "Measuring progress and value of ic implementation technology," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2016, pp. 1–8.

[73] K. Jo, S. Ahn, J. Do, T. Song, T. Kim, and K. Choi, "Design rule evaluation framework using automatic cell layout generator for design technology co-optimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 8, pp. 1933–1946, 2019.

[74] C. Lazzari, C. Santos, and R. Reis, "A new transistor-level layout generation strategy for static cmos circuits," in *2006 13th IEEE International Conference on Electronics, Circuits and Systems*. IEEE, 2006, pp. 660–663.

[75] X. Xu, N. Shah, A. Evans, S. Sinha, B. Cline, and G. Yeric, "Standard cell library design and optimization methodology for asap7 pdk," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 999–1004.

[76] B. Taylor and L. Pileggi, "Exact combinatorial optimization methods for physical design of regular logic bricks," in *Proceedings of the 44th annual Design Automation Conference*, 2007, pp. 344–349.

[77] P.-H. Wu, M. P.-H. Lin, T.-C. Chen, T.-Y. Ho, Y.-C. Chen, S.-R. Siao, and S.-H. Lin, "1-d cell generation with printability enhancement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 3, pp. 419–432, 2013.

[78] N. Ryzhenko and S. Burns, "Standard cell routing via boolean satisfiability," in *Proceedings of the 49th Annual Design Automation Conference*, 2012, pp. 603–612.

[79] C. J. Poirier, "Excellerator: Custom cmos leaf cell layout generator," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 8, no. 7, pp. 744–755, 1989.

[80] W. Ye, B. Yu, D. Z. Pan, Y.-C. Ban, and L. Liebmann, "Standard cell layout regularity and pin access optimization considering middle-of-line," in *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, 2015, pp. 289–294.

[81] M. Guruswamy, R. L. Maziasz, D. Dulitz, S. Raman, V. Chiluvuri, A. Fernandez, and L. G. Jones, "Cellerity: A fully automatic layout synthesis system for standard cell libraries," in *Proceedings of the 34th annual Design Automation Conference*, 1997, pp. 327–332.

[82] A. M. Ziesemer and R. A. da Luz Reis, "Simultaneous two-dimensional cell layout compaction using milp with astran," in *2014 IEEE Computer Society Annual Symposium on VLSI*. IEEE, 2014, pp. 350–355.

[83] Y.-L. Li, S.-T. Lin, S. Nishizawa, H.-Y. Su, M.-J. Fong, O. Chen, and H. Onodera, "Nctucell: A dda-aware cell library generator for finfet structure with implicitly adjustable grid map," in *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, 2019, p. 120.

[84] A. Sorokin and N. Ryzhenko, "Sat-based placement adjustment of finfets inside unroutable standard cells targeting feasible drc-clean routing," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*. ACM, 2019, pp. 159–164.

[85] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340.

[86] M. J. Rentmeesters, W. K. Tsai, and K.-J. Lin, "A theory of lexicographic multi-criteria optimization," in *Proceedings of ICECCS'96: 2nd IEEE International Conference on Engineering of Complex Computer Systems (held jointly with 6th CSESAW and 4th IEEE RTAW)*. IEEE, 1996, pp. 76–79.

[87] D. Park, D. Lee, I. Kang, C. Holtz, S. Gao, B. Lin, and C.-K. Cheng, "Grid-based framework for routability analysis and diagnosis with conditional design rules," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[88] C. Han, K. Han, A. B. Kahng, H. Lee, L. Wang, and B. Xu, "Optimal multi-row detailed placement for yield and model-hardware correlation improvements in sub-10nm vlsi," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 667–674.

[89] S. Sutanthavibul, E. Shragowitz, and J. B. Rosen, "An analytical approach to floorplan design and optimization," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 10, no. 6, pp. 761–769, 1991.

[90] L. Liebmann, G. Northrop, M. Facchini, L. R. Cazaux, Z. Baum, N. Nakamoto, K. Sun, D. Chanemougame, G. Han, and V. Gerousis, "Pre-PDK block-level PPAC assessment of technology options for sub-7nm high-performance logic," in *Design-Process-Technology Co-optimization for Manufacturability XII*, vol. 10588. International Society for Optics and Photonics, 2018, p. 1058808.

[91] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "Asap7: A 7-nm finfet predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016.

[92] L. Liebmann, J. Zeng, X. Zhu, L. Yuan, G. Bouche, and J. Kye, "Overcoming scaling barriers through design technology cooptimization," in *2016 IEEE Symposium on VLSI Technology*. IEEE, 2016, pp. 1–2.

[93] X. Xu, Y. Lin, V. Livramento, and D. Z. Pan, "Concurrent pin access optimization for unidirectional routing," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

[94] T. Stöhr, M. Alt, A. Hetzel, and J. Koehl, "Analysis, reduction and avoidance of crosstalk on vlsi chips," in *Proceedings of the 1998 international symposium on Physical design*, 1998, pp. 211–218.

[95] M. Martins, J. M. Matos, R. P. Ribas, A. Reis, G. Schlinker, L. Rech, and J. Michelsen, "Open cell library in 15nm freepdk technology," in *Proceedings of the 2015 Symposium on International Symposium on Physical Design*, 2015, pp. 171–178.

[96] P. Van Cleeff, S. Hougardy, J. Silvanus, and T. Werner, "Bonncell: Automatic cell layout in the 7-nm era," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2872–2885, 2019.

[97] D. Lee, D. Park, C.-T. Ho, I. Kang, H. Kim, S. Gao, B. Lin, and C.-K. Cheng, "SP&R: SMT-based Simultaneous Place-&-Route for Standard Cell Synthesis of Advanced Nodes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[98] S. M. Y. Sherazi, C. Jha, D. Rodopoulos, P. Debacker, B. Chava, L. Matti, M. G. Bardon, P. Schuddinck, P. Raghavan, V. Gerousis, A. Spessot, D. Verkest, A. Mocuta, R. H. Kim, and J. Ryckaert, "Low track height standard cell design in in7 using scaling boosters," in

Design-Process-Technology Co-optimization for Manufacturability XI, L. Capodieci and J. P. Cain, Eds., vol. 10148, International Society for Optics and Photonics. SPIE, 2017, pp. 279 – 286.

[99] S. Y. Sherazi, J. K. Chae, P. Debacker, L. Matti, P. Raghavan, V. Gerousis, D. Verkest, A. Mocuta, R. Kim, A. Spessot, and J. Ryckaert, "Track height reduction for standard-cell in below 5nm node: how low can you go?" in *Design-Process-Technology Co-optimization for Manufacturability XII*, J. P. Cain, Ed., vol. 10588, International Society for Optics and Photonics. SPIE, 2018, pp. 66 – 78.

[100] L. Matti, V. Gerousis, M. Berekovic, P. Debacker, S. M. Y. Sherazi, D. Milojevic, R. Baert, J. Ryckaert, R. han Kim, D. Verkest, and P. Raghavan, "Efficient place and route enablement of 5-tracks standard-cells through euv compatible n5 ruleset," in *Design-Process-Technology Co-optimization for Manufacturability XII*, J. P. Cain, Ed., vol. 10588, International Society for Optics and Photonics. SPIE, 2018, pp. 11 – 17.

[101] S. Sherazi, J. Chae, P. Debacker, L. Matti, D. Verkest, A. Mocuta, R. Kim, A. Spessot, A. Dounde, and J. Ryckaert, "Cfet standard-cell design down to 3track height for node 3nm and below," in *Design-Process-Technology Co-optimization for Manufacturability XIII*, vol. 10962. International Society for Optics and Photonics, 2019, p. 1096206.

[102] J. Smith, "Design Technology Co-Optimization Approaches for Integration and Migration to CFET and 3D Logic," in *THE SURFACE PREPARATION AND CLEANING CONFERENCE (SPCC)*. Linx, 2019.

[103] C.-K. Cheng, C.-T. Ho, D. Lee, and D. Park, "A Routability-Driven Complimentary-FET (CFET) Standard Cell Synthesis Framework using SMT," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2020, pp. 1–8.