UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MICROELETRÔNICA

JUCEMAR LUIS MONTEIRO

Algorithms to Improve Area Density Utilization, Routability and Timing During Detailed Placement and Legalization of VLSI Circuits

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Microeletronics

Advisor: Prof. Dr. Marcelo de Oliveira Johann
Coadvisor: Prof. Dr. Laleh Behjat

Porto Alegre
May, 17^{th} 2019

CIP — CATALOGING-IN-PUBLICATION

Monteiro, Jucemar Luis

Algorithms to Improve Area Density Utilization, Routability and Timing During Detailed Placement and Legalization of VLSI Circuits / Jucemar Luis Monteiro. – Porto Alegre: PGMICRO da UFRGS, 2019.

246 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Microeletrônica, Porto Alegre, BR–RS, 2019. Advisor: Marcelo de Oliveira Johann; Coadvisor: Laleh Behjat.

1. Microelectronic. 2. EDA. 3. Placement. 4. Placement Optimization. 5. Network Flow. 6. Branch and Cut. I. Johann, Marcelo de Oliveira. II. Behjat, Laleh. III. Título.

"If I have seen further it is by standing on the shoulders of Giants."

— Sir Isaac Newton

Esta tese é dedicada aos meus pais, Juvencio e Gilvite,

e ao meu avô, Angelo Baldissera.

This thesis is dedicated to my parents, Juvencio and Gilvite,

and my grandparent, Angelo Baldissera.

ACKNOWLEDGEMENT

To my parents.

To my supervisors Dr. Marcelo Johann and Dr. Laleh Behjat

To the evaluation committee members Dr. Andrew Kahng, Dr. Patrick Groeneveld and Dr. Ismail Bustany.

To the UFRGS colleagues Guilherme, Mateus, Éder, Henrique, Isadora and Gracieli.

To the University of Calgary and ELAP program for the researching exchange scholarship.

To Nima for the collaboration.

To Synopsys and Cadence for the internship opportunities.

To the CAPES for the Ph.D. scholarship.

The thesis work had soundtracks composed by Epica, Within Temptation, Delain, Amaranthe, AC/DC, Black Sabbath, Iron Maiden, Scorpions, Taylor Davis, and Lindsey Stirling.

ABSTRACT

Placement is a challenging stage in the Very Large-Scale Integration (VLSI) physical design flow. In modern VLSI designs, several design restrictions have been imposed to address the complexity of advanced Complementary Metal-Oxide Semiconductor (CMOS) fabrication nodes. Design restrictions have a considerable influence on achieving the optimized circuit solution. The quality of the placement solution has a significant impact on circuit performance. In placement, achieving circuit requirements of timing and routability is a very challenging task. Timing and routability requirements are especially hard to achieve in circuits which have regions with high-density area utilization. Moreover, the quality of placement has a direct influence on circuit quality and optimization effort of Clock Tree Synthesis (CTS), routing, and post-placement algorithms. In this thesis, the first contribution is an incremental timing-driven placement algorithm subject to routability. The proposed timing-driven placement algorithm relies on net and path characteristics to compute optimized-timing cell positions. Optimized-timing positions are accepted only if these positions are inside regions free of routing violation. The second contribution is a cell spreading algorithm. The objective is to move cells out of high-density regions considering adverse side effects on moved cells. The proposed cell spreading algorithm relies on network flow and branch and cut techniques to minimize high-density regions. Area flows are moved from high-density to low-density regions with optimized cost paths. Therefore, cell concentration is reduced, and white spaces are opened in high-density regions with minimized adverse side effects on moved cells. Legalization, detailed placement, and post-placement algorithms can use these white spaces to further optimize the placement solution. In high-density regions, white spaces are limited resources. In the traditional placement flow (global placement, legalization, and detailed placement), the placement optimization is limited by the strict placement flow. The proposed cell spreading algorithms can be integrated into a mixed placement flow that is composed of interleaved legalization and detailed placement algorithms. In this mixed placement flow, the restriction to optimize detailed placement in a legalized netlist can be relaxed. Detailed placement algorithms can achieve further placement optimization with less restricted placement formulation. The focus of legalization algorithms can only be to fix cell overlap with minimized adverse effects on placement, instead of also fix density area violation. The proposed cell spreading algorithm is applied in legalization and detailed placement stages to optimize area density utilization. The proposed legalization algorithm has achieved improvement on average (30%) and maximum (350%) cell displacement compared to the state of the arts legalization algorithms. In detailed placement, the proposed algorithm has been evaluated in industrial and academic placement flows. In industrial placement flow, the proposed cell spreading algorithm has achieved improvement in cell displacement, power consumption, and timing. The proposed cell spreading algorithm can improve the quality of placement in mixed placement flow in both industrial and academic environments. The proposed algorithm provides a uniform cell distribution placement in constrained designs with minimized adverse side effects on moved cells.

Keywords: Microelectronic. EDA. Placement. Placement Optimization. Network Flow. Branch and Cut.

Algoritmos para Aprimorar Densidade de Utilização de Área, Rotabilidade e Tempo de Propagação Durante o Posicionamento Detalhado e a Legalização de Circuitos VLSI

RESUMO

O posicionamento é um estágio desafiante no fluxo de projeto físico para integrar circuitos VLSI (sigla do inglês *Very Large-Scale Integration (VLSI)*). Em projetos modernos de circuitos VLSI, diversas restrições de projeto são impostas para visar a complexidade dos avançados nodos de fabricação CMOS (sigla do inglês). As restrições de projeto têm uma considerável influência em obter soluções otimizadas de circuitos. A qualidade do posicionamento tem uma significativa influência no desempenho do circuito. No posicionamento, obter os requisitos do circuito em tempo de propagação e rotabilidade é uma tarefa desafiante. Requerimentos de tempo de propagação e rotabilidade são especialmente difíceis de obter em circuitos que tem regiões com alta densidade de utilização de área. Além disso, a qualidade do posicionamento tem influência direta na qualidade do circuito e no esforço de otimização dos algoritmos de síntese da árvore de relógio (sigla do inglês CTS), roteamento e pós-posicionamento otimização. A primeira contribuição apresentada nessa tese é um algoritmo incremental de posicionamento para otimizar violações no tempo de propagação sujeito a rotabilidade. O algoritmo proposto é baseado em características das redes e dos caminhos de dados para computar posições otimizadas para as células. Posições otimizadas são aceitas somente se elas estão dentro de regiões livres de violações de roteamento. A segunda contribuição apresentada nessa tese é um algoritmo de espalhamento de células. O objetivo é mover células fora de regiões com alta densidade de área considerando efeitos adversos nas células movidas. O algoritmo proposto é baseado em técnicas de *network flow* e *branch and cut* para minimizar regiões com alta densidade de área. Fluxos de área são movidos de regiões com alta densidade de área para regiões com baixa densidade de área com caminhos com custo otimizado. Assim sendo, a concentração de células é reduzida e espaços são abertos em regiões com alta densidade de área com minimizado efeitos adversos nas células movidas. Algoritmos de legalização, posicionamento detalhado e pós-posicionamento podem utilizar esses espaços abertos para otimizar mais a solução de posicionamento. Em regiões com alta densidade de área, espaços são recursos limitados. No fluxo de posicionamento tradicional (posicionamento global, legalização e posicionamento detalhado), a otimização do posicionamento é limitada pelo fluxo tradicional. O algoritmo proposto de espalhamento de células pode ser integrado em um fluxo misto de posicionamento que é composto de algoritmos intercalados de legalização e posicionamento detalhado. Nesse fluxo de posicionamento misto, a restrição para otimizar posicionamento detalhado em uma *netlist* legalizada pode ser relaxada. Algoritmos de posicionamento detalhado podem obter melhor posicionamento com uma formulação de posicionamento menos restrita. O foco dos algoritmos de legalização pode ser somente remover sobreposição de células com efeitos adversos minimizados no posicionamento ao invés de também minimizar violações de densidade de utilização de área. O algoritmo proposto de espalhamento de células é aplicado nos estágios de legalização e posicionamento detalhado para otimizar densidade de utilização de área. O algoritmo proposto de legalização obteve melhoria no espalhamento médio (30%) e máximo (350%) de células comparado com algoritmos de legalização estado da arte. No posicionamento detalhado, o algoritmo proposto foi avaliado em fluxos de posicionamento industrial e acadêmico. No fluxo de posicionamento industrial, o algoritmo proposto melhorou espalhamento de células, potência dissipada e tempo de propagação. O algoritmo proposto de espalhamento de células pode melhorar a qualidade do posicionamento em fluxos de posicionamento mistos em ambiente industrial e acadêmico. O algoritmo proposto fornece posicionamento com distribuição uniforme de células em projetos limitados com efeitos adversos minimizados nas células movidas.

Palavras-chave: EDA, Posicionamento, Otimização, *Network Flow*, *Branch and Cut*.

LIST OF ABBREVIATIONS AND ACRONYMS

ABU Average Bin Utilization

ACD Average Cell Displacement

ACM Association for Computing Machinery

ASIC Application-Specific Integrated Circuit

AT Arrival Timing

B2B Bound to Bound

BFS Breadth-First Search

BIST Built-In Self-Test

CCS Composite Current Source

CMOS Complementary Metal-Oxide Semiconductor

CPPR Common Path Pessimism Removal

CPU Central Processing Unit

CTS Clock Tree Synthesis

DAC Design Automation Conference

DEF Design Exchange Format

DRAM Dynamic Random-access Memory

ECSM Effective Current Source Model

EDA Electronic Design Automation

Eh?L Eh?Legalizer

ELAP Emerging Leaders of the Americas Program

FPGA Field-Programmable Gate Array

FPL FastPlace Legalizer

FSM Finite-state Machine

GCC GNU Compiler Collection

GCell Global Cell

GND Ground

GPU Graphics Processing Unit

GRO Global Routing Overflow

HDL Hardware-Description Language

HPWL	Half Perimeter Wire Length
IC	Integrated Circuit
ICCAD	International Conference on Computer Aided Design
ILP	Integer Linear Programming
IO	Input/Output
IP-Core	Semiconductor Intellectual Property Core
ISCAS	International Symposium on Circuits and Systems
ISPD	International Symposium on Physical Design
ITDP	Incremental Timing-Driven Placement
LAL	Look Ahead Legalization
LCB	Local Clock Buffer
LEF	Layout Exchange Format
LP	Linear Programming
MIP	Mixed Integer Programming
NDR	Non Default Rule
NFCS	Network Flow-based Cell Spreading
NFL	Network Flow-based Legalization
NLDP	Non Linear Delay Model
NPA	Non Placeable Area
NP	Nondeterministic Polynomial Time
OAL	Obstacle-aware Legalization
OF	Overfilled
P	Polynomial Time
PATMOS	International Workshop on Power And Timing Modeling, Optimization and Simulation
PCB	Printed Circuit Board
PEKO	Placement Examples with Known Optimal
PLL	Phase-locked Loop
PTM	Predictive Technology Model
PVT	Process, Voltage, and Temperature
QoR	Quality of Results

RAITDP Routing-aware Incremental Timing-driven Placement

RAT Required Arrival Timing

RMST Rectilinear Minimum Spanning Tree

RO Routing Overflow

RSA Rectilinear Steiner Arborescence

RSMT Rectilinear Steiner Minimum Tree

RTL Register-Transfer Level

SA Simulated-Annealing

SPEF Standard Parasitic Exchange Format

SRAM Static Random-Access Memory

STA Static Timing Analysis

STST Single-Trunk Steiner Tree

TAU International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems

TNS Total Negative Slack

TQoR Timing Quality of the Results

VDD Power

VIA Vertical Interconnect Access

VLSI Very Large-Scale Integration

WNS Worst Negative Slack

LIST OF SYMBOLS

$\|x\|$	Absolute Value
α	Alpha
β	Beta
C	Capacitance
(x, y)	Cartesian point
$\{\}$	Curly Brackets
$°\mathrm{C}$	Degree Celsius
Δ	Delta
$\frac{x}{y}$	Division
$\exists$	Exist an element
$=$	Equal Symbol
$\forall$	For all elements
$f(x)$	Function Sign
γ	Gamma
$>$	Greater Than
$\geq$	Greater Than or Equal to
$\in$	In Symbol
∞	Infinity
$\mathbb{I}$	Integer Numbers
$<$	Less Than
$\leq$	Less Than or Equal to
m	Meter
m	Milli
µ	Micro

mod	Modulo Operation
$\times$	Multiplication
n	Nano
$\mathbb{N}$	Natural Numbers
$-$	Negative or Subtraction Sign
ω	Omega
$()$	Parenthesis
$\%$	Percentage
π	Pi
$+$	Positive or Addition Sign
$\mathbb{R}$	Real Numbers
RC	Resistance-Capacitance
$\rightarrow$	Right Arrow
s	Second
$\sqrt{x}$	Square Root
$\{x, y\}$	Set Sign
$\sum$	Summation
τ	Tau
$\overrightarrow{x}$	Vector representation
$\vert$	Vertical Slash
V	Volt

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

The fast advancements in Complementary Metal-Oxide Semiconductor (CMOS) technology has allowed an increase in the density of transistors and to integrate more features in Integrated Circuits (ICs). Advanced CMOS technologies imposes a significant number of design rules to improve manufacturability. In digital design flow, new design rules restrict the search space for optimization algorithms. The fast increase in density of transistors and shrinking of CMOS technology imposes new optimization challenges to the digital design flow. Modern circuits may have millions of cells and billions of transistors. These modern circuits may also be composed of thousands of macroblocks. For example, a 32 nm Intel Core i7 (INTEL..., 2018) microprocessor has 2.7 billions transistors (KURD et al., 2010).

Digital design flow is a sequence of steps to elaborate and to synthesize digital circuits. Design flow is split into 1) circuit synthesis, 2) simulation and formal verification, and 3) implementation of test structures. In design flow, the initial step is to elaborate the circuit requirement document. Then, circuit micro-architecture is designed. Circuit behavior is coded in Register-Transfer Level (RTL) description. Logic functions in RTL are optimized in the logic synthesis stage. In this synthesis stage, the optimized circuit is mapped to library cells. In the last stage (physical synthesis), the circuit layout is built and optimized. Circuit floorplan is established. Cells are placed in optimized positions, and pins are connected with Vertical Interconnect Accesses (VIAs) and wire segments. Physical synthesis stage is split into 1) *floorplanning*, 2) *placement*, 3) *Clock Tree Synthesis (CTS)*, and 4) *routing*. The traditional placement flow is divided into 1) *global placement*, 2) *legalization* and 3) *detailed placement*. Simulation and formal verification are performed in parallel to the synthesis flow. The objective simulation and formal verification is to detect logic errors in netlist and layout. A second flow is also performed in parallel to design and to implement test structures.

Digital design flow is composed of several optimization algorithms. These algorithms rely on heuristics and formal methods to optimize circuit objectives subject to constraints. In global placement, optimized rough cell positions are computed to minimize total wire length subject to area density utilization. Circuit timing and routability requirements are directly or indirectly considered while optimizing global placement. Cell overlapping and cell alignment to rows are relaxed in global placement. In legalization, cells are placed in positions where cells are aligned to rows and are free of cell overlapping. Finally, in detailed placement, the circuit is locally optimized. One cell or a small set of cells are moved to optimized positions in each algorithm iteration. The common objectives to optimized in detailed placement algorithms are 1) wire length, 2) power consumption, 3) timing violations, 4) routability, and 5) manufacturability.

Optimizing timing and routability in placement is a challenging task. Timing and routability requirements are especially hard to be achieved in circuits that have regions with high-density area utilization. Usually, timing and routability optimization algorithms relax routability and timing constraints, respectively. The quality of the placement solution has a direct impact on the quality of the circuit solution and the optimization effort of the CTS, routing, and post-placement algorithms.

Placement may be stated as an optimization problem with objectives and constraints. The typical placement objectives are 1) minimizing total wire length, and 2) optimizing area density utilization (KIM; LEE; MARKOV, 2012). Typical placement constraints are 1) maximum area density utilization, 2) maximum signal delay, 3) maximum routing resources, and 4) design rules. Placement algorithms rely on heuristics and mathematical formulations to optimize placement solution. Placement constraints impose limits to the search space to optimize the placement solution (ALPERT et al., 2012; MARKOV; HU; KIM, 2015).

1.1 Motivation

Placement is a hard problem to obtain an optimized solution. Optimization placement algorithms depend heavily on heuristics and formal methods. Advanced CMOS technologies impose new design restrictions which must be observed. Placement problem formulation and design restrictions open opportunities to research heuristics and formal methods for placement algorithms to improve placement solution.

In detailed placement, achieving an optimized placement solution in high-density regions is a challenging task. In high-density regions, the lack of white spaces severely limits to improve the quality of placement solution. On the other hand, opening white spaces in high-density regions can be a costly operation regarding the adverse side effects in the placement and required computing resources. A rough cell spreading algorithm can lead to significant adverse side effects on moved cells. Circuit constraints may be very challenging to achieve with a rough cell spreading procedure. Therefore, a cell spreading algorithm with minimized adverse side effects on moved cells can aid placement optimization algorithms indirectly. Newly opened white spaces can be used by these optimization algorithms to further improve the placement solution.

In the detailed placement stage, optimization algorithms compute optimized-positions to move cells. In optimized-positions, white spaces are required to place the cells. Detailed placement algorithms could improve the placement solution if white spaces are available in high-density regions. Moreover, the traditional placement flow (global placement, legalization and detailed placement) limits placement optimization. Usually, detailed placement algorithms require the input netlist to be legalized. These detailed placement algorithms commonly provide legal-optimized placement solution. Several detailed placement algorithms may optimize placement with relaxed legalization constraint. Therefore, the placement solution can be further improved. Legalization algorithms with minimized cell displacement can later perform circuit legalization.

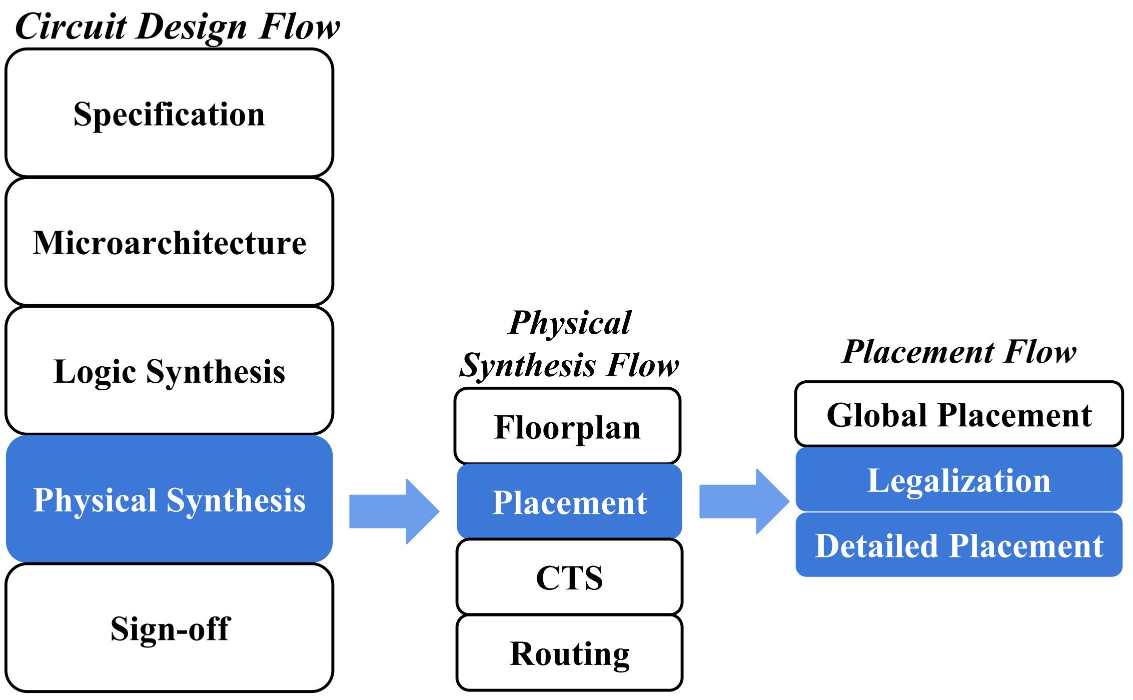
The fast advances in CMOS technology imposes more restrictions and limits search space to achieve optimized placement solution. Placement algorithms require complex heuristics and advanced optimization techniques to improve restricted placement formulations. Placement objectives and constraints could be addressed together while moving cells to optimized positions. Extra computing resources are required to achieve this optimized placement solution. Therefore, placement algorithms which can optimize placed netlist with several objectives, numerous constraints, and minimized computing resources are required. These algorithms can further improve the placement solution by smartly exploring the relation of objectives and constraints. A cell spreading algorithm could provide a uniform cell distribution with minimized side effects on moved cells. Therefore, detailed placement algorithms can focus to optimize objectives with relaxed area density constraint.

1.2 Contributions

In this thesis, the four main contributions are listed as following.

- **Development of a routing-aware incremental timing-driven detailed placement algorithm.** Detailed placement algorithms either optimize timing or routability. In this thesis, a timing-driven detailed placement algorithm subject to routability is proposed. In the proposed algorithm, optimized-timing cell positions are computed subject to routing overflow restriction.

Figure 1.1: Digital design flow. In this thesis, the proposed contributions are to optimized placement solutions in legalization and detailed placement stages



Source: Author (2019).

- **Optimization of area utilization through cell spreading algorithm**. A cell spreading algorithm using network flow and branch and cut techniques is proposed. Optimized-cost paths are computed using these techniques to move cells out from high-density regions. A cell displacement cost model in which the history and direction of cell movements are integrated into the proposed cell spreading algorithm. In this cost model, the direction of cell movements close to or far away from initial positions is easily obtained. Cells which are going to be moved closer to initial positions further improve cell displacement. Otherwise, cell displacement is increased.
- **Development of a legalization algorithm using the proposed cell spreading technique.** The network flow-based cell spreading technique is used in the proposed network flow-based legalization algorithm. The proposed cell spreading algorithm is used to optimize area density utilization. In cell legalization, optimized-legal cell positions are also searched in the neighboring rows. Combined cell spreading and cell legalization techniques allow for the achievement of further cell displacement optimization.
- **Optimization of area density utilization in detailed placement using the proposed cell spreading technique.** The cell spreading technique is used to optimize area density utilization in detailed placement. Cells are moved out from high-

density regions subject to maximum cell displacement constraint. The proposed algorithm opens white spaces in high-density regions. These white spaces can be used by other detailed placement algorithms to further achieve circuit improvement. In the cell spreading procedure, cell movements can be subject to several restrictions (e.g., timing, and routability). The proposed algorithm has been implemented in commercial and academic environments. In both environments, the proposed algorithm has contributed to further improve the placement solution.

1.3 Thesis Organization

This thesis is organized with this introductory chapter and eight chapters summarized as following.

Chapter 2: In chapter 2, an introduction of digital circuits and digital design is presented. The chapter introduction is given in Section 2.1. In Section 2.2, circuit definitions are presented. Static Timing Analysis (STA) and electrical wire models to estimate timing propagation of electric signals are presented in Sections 2.3 and 2.4, respectively. Electric signal delay is essential data to guide optimization algorithms in logic and physical synthesis. Digital design flow is introduced in Section 2.5. Digital design flow is a sequence of steps to *elaborate circuit requirements*, to *implement circuit microarchitecture*, to *code* and to *synthesize* netlist. Finally, in Section 2.6, the chapter summary is shown.

Chapter 3: In this chapter, Electronic Design Automation (EDA) concepts and correlated placement algorithms and techniques are introduced. In Section 3.1, the chapter introduction is shown. Algorithm and graph definitions are briefly presented in Sections 3.2 and 3.3, respectively. Optimization techniques, especially network flow and branch and cut are introduced in Section 3.4. In Section 3.5, net models to decompose hyperedge nets for placement optimization algorithms are shown. In Section 3.6, metrics to evaluate the quality of placement solution are presented. Grid graph of bins to compute optimized paths in the proposed cell spreading algorithms is presented in Section 3.7. In Section 3.8, physical synthesis in terms of EDA algorithms is given. In this Section, synthesis flow, algorithms, and definitions are also presented. Finally, the summary of Chapter 3 is shown in Section 3.9.

Chapter 4: In this chapter, digital circuit placement is introduced. In Section 4.1, the chapter introduction is given. In Section 4.2, global placement, global placement techniques, and placement objectives are discussed. In Section 4.3, legalization and network flow-based legalization are presented. In Section 4.4, detailed placement, detailed placement techniques, and detailed placement objectives are shown. Relevant and recent placement algorithms are highlighted in Sections 4.2, 4.3, 4.4. Finally, the chapter summary is given in Section 4.5.

Chapter 5: In this chapter, the proposed algorithm to move critical timing cells to optimized timing positions subject to routability is presented. In Section 5.1, the chapter introduction is given. The proposed algorithm optimization flow is presented in Section 5.2. Experimental results are discussed in Section 5.3. Finally, conclusions are given in Section 5.4.

Chapter 6: In this chapter, the proposed network flow-based cell spreading algorithm is presented. In the proposed algorithm, network flow and branch and cut techniques are integrated. The objective is to compute minimum cost paths to move cells out from high-density regions with minimized adverse side effects. Optimized-cost paths are searched in an n-ary tree. Branches of the tree are opened only if the total cost of ancestor's bins is lower than the upper limit cost. The upper limit cost is the cost of the current minimum cost path. This current path is replaced every time a new path with lower cost is found. In Section 6.1, the chapter introduction is given. In Section 6.2, insights into the proposed network flow-based cell spreading algorithm are presented. In Section 6.3, non-overlapping, and overlapping types of grid graphs are given. In Section 6.4, the proposed cell spreading algorithm is discussed. Chapter summary is given in Section 6.5. The proposed cell spreading algorithm may be applied in legalization and detailed placement with minor adjustments. In this chapter, only the core of the proposed algorithm without experimental results is given. Applied cell spreading technique in legalization and detailed placement are presented in Chapters 7 and 8, respectively.

Chapter 7: The proposed cell spreading algorithm, which is shown in Chapter 6, is applied in the legalization stage. In this chapter, the proposed Network Flow-based Legalization (NFL) algorithm is presented. In the first stage, high-density regions are minimized with a cell spreading algorithm. Cells are moved out from high-density with branch and cut, and a network flow-based path augmentation algorithm. Optimized-cell positions are computed considering the history and direction of cell movements. Optimization of high-density regions is limited by small dimensions of bins and restriction to overlap bins with macroblocks and fixed cells. In the second stage, cells are placed in legal positions. Legal positions are also searched in neighboring rows. In Section 7.1, the chapter introduction is given. In Section 7.2, correlated network flow-based legalization algorithms are presented. In Section 7.3, the proposed network flow-based legalization algorithm is shown. Experimental results are discussed in Section 7.4. Finally, the chapter summary is presented in Section 7.5.

Chapter 8: The proposed cell spreading algorithm, which is presented in Chapter 6, is applied in the detailed placement stage. In this chapter, the proposed Network Flow-based Cell Spreading (NFCS) algorithm is presented. Cells are moved out from high-density regions with minimized cell displacement cost. In the cell displacement cost model, optimized cost positions are computed considering the direction and the history of cell movements. This cell spreading algorithm has larger bin dimensions and relaxed restrictions of bin overlapping with macroblocks and fixed cells. In Section 8.1, the chapter introduction is given. In Section 8.2, the proposed network flow-based cell spreading algorithm is presented. Experimental results are shown in Section 8.3. Finally, chapter conclusions are given in Section 8.4.

Chapter 9: In this chapter, conclusions are given. In Section 9.1, the list of main contributions of this thesis are presented. Potential future research directions are given in Section 9.2. Reference of published journal and conference papers, awards, and the open-source framework are shown in Section 9.3.

2 DIGITAL CIRCUIT DESIGN

2.1 Introduction

The design of digital circuits is a challenging task. Modern circuit design strongly depends on EDA tools. Digital designers use and rely on software tools intensively to implement, optimize, verify, simulate, and analyze digital circuits. Design stages, such as circuit synthesis, verification, and simulation, require specialized EDA tools.

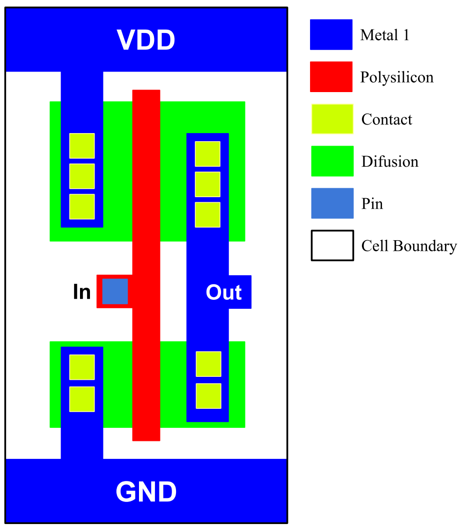
In this chapter, relevant term definitions of circuits and circuit design are presented. This chapter also gives a global view of digital circuits and digital design flow. Circuits and digital design are associated with a vast amount of terms and keywords. Several of these terms and keywords may be unknown, confusing or ambiguous. In this chapter, direct and clear definitions of the main circuits and digital circuit design terms are given. These term definitions aim to avoid misunderstanding in the following thesis' chapters. Digital design flow is a complex and vast set of synthesis and optimization stages. This chapter also gives a foundation of circuit term definitions, static timing analysis, electrical wire models and digital design flow. In this chapter, term definitions, analysis, and estimation models and digital design flow are presented in the bottom-up fashion. Initially, essential components are presented. These components are used in intricate flows, and models to be integrated in complex circuits.

This chapter is organized as follows: In Section 2.2, definition terms of digital circuits are given. The main concepts of static timing analysis are presented in Section 2.3. In Section 2.4, common electric wire models to estimate electric signal characteristics are highlighted. Digital design flow is presented in Section 2.5. Finally, the chapter summary is given in Section 2.6.

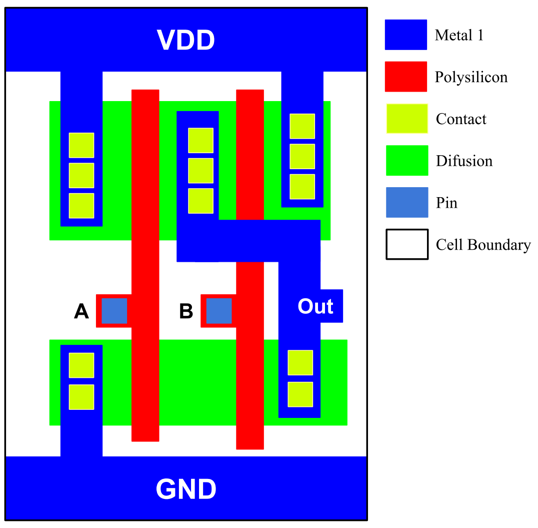
2.2 Digital Circuit Definitions

In this section, term definitions of digital circuits are given. These term definitions are grouped in sets related to parts of circuits and digital circuit design.

Figure 2.1: Layout of inversor (a) and NAND (b) standard cells. Layout is a set of rectangular geometries that abstract electric functions. Standard cells are implementations of Boolean functions



(a) Inversor Standard Cell Layout



(b) NAND Standard Cell Layout

Source: Author (2019).

2.2.1 Library Cells

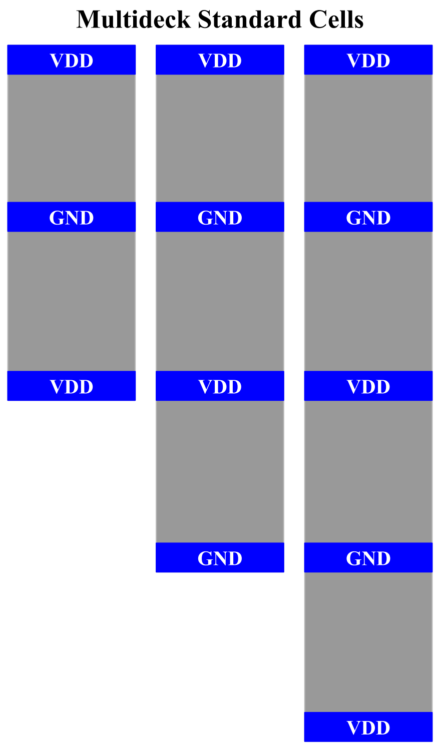
Digital circuits are composed of an enormous amount of electronic components which are synthesized and optimized. Electric circuit components are transistors, capacitors, resistors, and inductors. *Transistors* are switches and amplifiers of electric signals. Transistors are enabled or disabled by changing the voltage level in the gate terminal. Functional components of digital circuits are transistors.

Standard cells are Boolean functions implemented in semiconductor material with transistors. In Figures 2.1a and 2.1b, layouts of inversor and NAND standard cells are presented. In digital circuits, standard-cells are designed with a predefined standard height. The standard height is approximately the width of ten routing tracks. Standard-cells that have height length higher than standard height are called *multideck or multirow standard-cells*. In Figure 2.2, multideck standard cells are presented.

Power rails are Power (VDD) and Ground (GND) tracks in standard-cell layout. In the standard-cell layout, power rails are placed a row height distance for each other. In digital circuits, cells that are horizontal neighbors of each other can share the same power rails. Standard-cells have $n + 1$ power tracks, where $n = \frac{cell\ height}{row\ height}$ and $n > 0$.

Standard-cell libraries are a set of standard-cell layouts that are electrically characterized in CMOS process nodes. Different layout versions of standard-cells may be

Figure 2.2: Multideck standard cells have cell height higher than default cell height. These standard cells have overlap with two or more rows



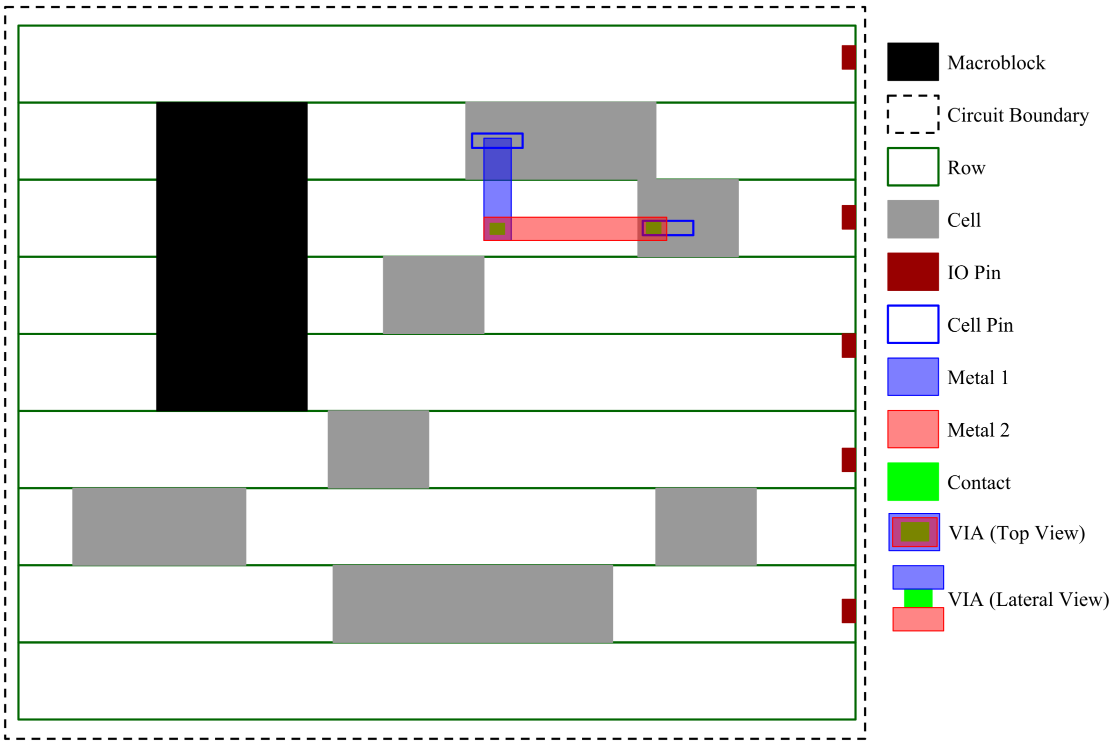
Source: Author (2019).

available in standard-cell libraries. The layout of standard-cells may be modified according to driver strength (gate width), power leakage, specific applications such as clock buffers, and test structures. There are several versions of standard-cell layouts for the same Boolean function. In modern technology libraries, standard-cells' height may be higher than the standard height. In this case, the remainder of the modulo operation of the cell and row heights ($cell\ height \bmod row\ height$) must be equal to zero.

2.2.2 Digital Circuit

In digital circuits, optimized-logic functions are mapped to standard-cells. Cells are instances of standard-cells that logic functions have been mapped. The same standard-cell may be replicated numerous times in digital circuits. Digital circuits are composed of thousands to millions of cells and up to thousands of macroblocks. *Die* is a small block of semiconductor material where electronic components are manufactured. *Die size* is determined by the necessary space to place circuit cells and macroblocks. In some digital circuits, extra space is required to place pads. *Circuit core* is the area of the die to place cells and macroblocks. Pads are placed inside the die area but outside of the circuit core. Pads form a ring around the circuit core. Die and circuit cores have the same area and shape in digital circuits without pads. In Figure 2.3, an example of a digital circuit is presented.

Figure 2.3: An example of a digital circuit. Digital circuit may be composed of macroblocks, cells, IO pins or pads, wires and VIAs. Cells are placed inside circuit core boundaries. These cells are aligned to row and site boundaries



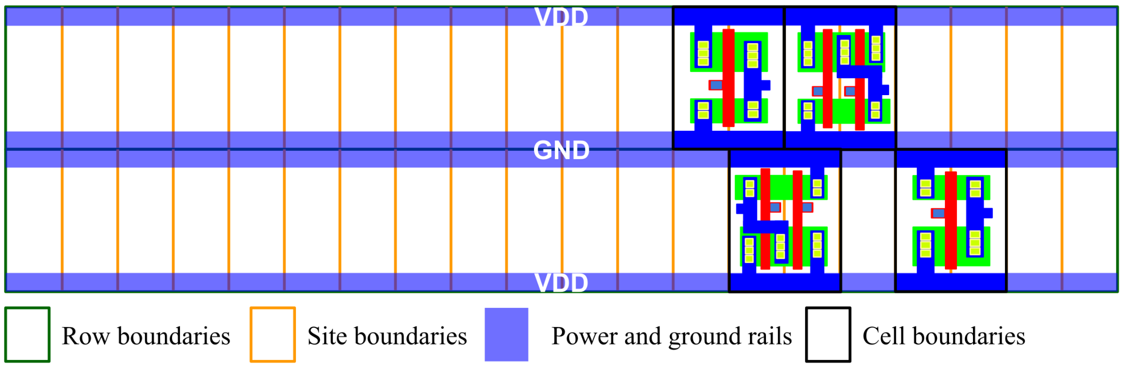
Source: Author (2019).

Module is part of the circuit that comprises a set of cells. A module can also be denominated as *submodule*. *Macrocells*, *macroblocks* or *blocks* are modules that have predefined function and fixed dimensions (e.g. Static Random-Access Memory (SRAM), Dynamic Random-access Memory (DRAM), Central Processing Unit (CPU), hardware accelerator, and so forth). Usually, macroblocks contain a considerable number of transistors, and macroblocks have dimensions which are significantly higher than standard-cells.

Net is composed of a set of pins that have the same electric potential. *Supply nets* are logic connections of power supply pins (VDD and GND). *Power pins* are connected with metal segments and VIAs. Digital circuits are described in netlist format. *Netlist* is a circuit description of nets which are connected to pins of cells or circuit IO interface. In mapped netlists, cells have reference to standard-cells.

A *Row* is a predefined region of the circuit core. The height of the row is also equal to the standard height of cells. Rows are partitioned into a set of sites. Row sites are small rectangular regions of rows. Cells inside a row share the same horizontal VDD and GND tracks. Neighboring rows can also share the same VDD and GND rails. In neighboring rows, cells must be vertically flipped to share correctly the power rails. In Figure 2.4, circuit rows are presented. In digital circuits, instances of multideck standard-cells are called *multideck cells*.

Figure 2.4: A row is a small part of the circuit core. Rows are divided into site areas. Cells must be aligned to the left, bottom and top site boundaries. Rows also share power rails with neighboring rows. Power rails are parallel and horizontal metal track to provide VDD and GND to cells



Source: Author (2019).

Circuits, cells, and macroblocks have external interfaces which are called pins or pads. In digital circuits, pins are also denominated as *ports*. Pins and pads are electric terminals to connect internal circuit components to the external environment. Pins are the interface to connect cells or macroblocks inside of another digital circuit. Pins are connected with metal segments which are routed in metal layers. Digital circuits that contain only pins which are external interface are called *Semiconductor Intellectual Property Cores (IP-Cores)*. Pads are interfaces that connect digital circuits externally to the package (encapsulation). Encapsulated circuits are integrated in Printed Circuit Boards (PCBs).

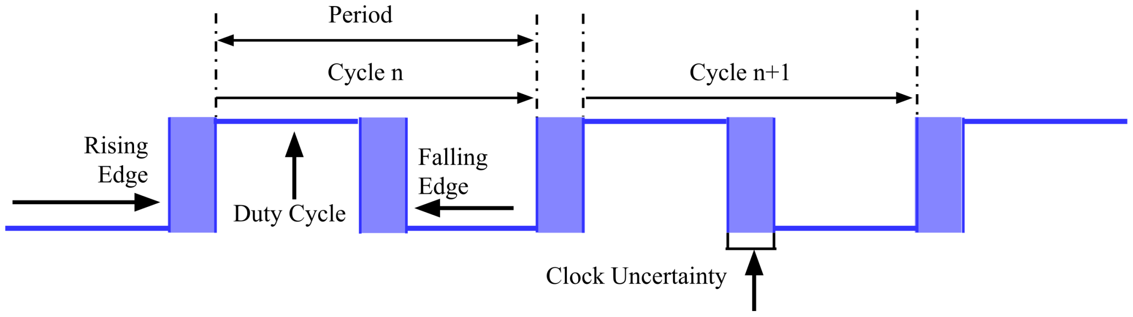
2.2.3 Manufacturing Process

The manufacturing process is a sequence of material transformation in the semiconductor substrate. In the transformation process, patterns of circuit layout are transferred to the semiconductor material. The circuit layout is the result of synthesis and optimization. *Layout* is a set of rectangular shapes which determine mask patterns. *Masks* are used in the CMOS manufacturing process to transfer layout geometries to a substrate. Circuit fabrication is a process divided into several *manufacturing layers*. Manufacturing layer is a specific part of the fabrication process to manufacture active components (transistors) and connections (metal layers). Usually, in the manufacturing process, transistors are associated with polysilicon and active layers (diffusion), and interconnections are associated with poly and metal layers. *Contact* is a direct connection from the active layer (transistor) to polysilicon or metal layers. *VIAs* are metal pieces that are used to connect two metal segments in adjacent metal layers. Metal segments are separated with an insulator. In each net, pins are connected with metal segments and VIAs.

2.2.4 Clock Signal

Digital circuits require a periodic electric signal to control and to synchronize the correctness of operations of cells and macroblocks. In digital circuits, a *clock* is a periodic signal that synchronizes and controls operations of registers, memories, Finite-state Machines (FSMs), and pipelines. In Figure 2.5, an example of the clock of digital circuits is presented.

Figure 2.5: In digital circuits, the clock is a periodic square wave. The clock signal controls and synchronizes operations of sequentical circuit elements



Source: Author (2019).

The clock is an infinite sequence of electric signals that transition in opposite directions (rise and fall, or fall and rise) in an interval of time. In rising transitions (edges), the voltage of the clock is changed from GND to VDD levels. In falling transitions (edges), clock voltage is changed from VDD to GND levels. A *clock pulse* is a rise or fall transition. Consecutive clock transitions occur periodically. A *clock period* is the total time between two clock transitions in the same direction (*RISE*-fall-*RISE* or *FALL*-rise-*FALL*). *Duty cycle* is the total time in which the voltage of the clock signal is in VDD level for each clock period.

Clock duty cycle is defined by the pair of rising and falling transitions in each clock period. A *clock cycle* is the sequential repetition of clock periods. The clock period is the main component of the maximum time that data signals have to propagate from start to end points. *Clock jitter* is the time difference between two consecutive clock transitions. Clock jitter originates from Phase-locked Loop (PLL) and the crystal that generates clock pulses. *Clock skew* is the difference of clock arrival time in two distinct clock pins of sequential components. Clock skew is the difference of the clock signal delay in clock tree branches. *Clock uncertainty* is the summation of clock jitter and clock skew.

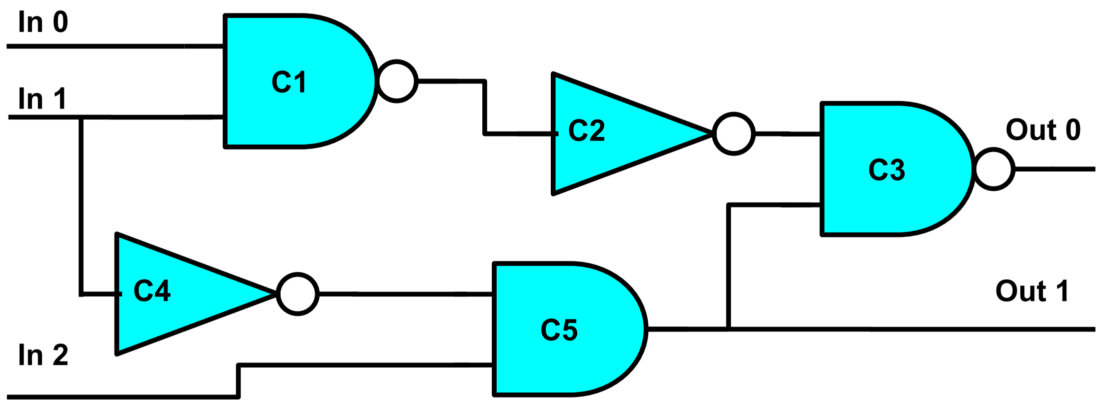
2.2.5 Memory Components

A *memory element* is a circuit to store data. In digital circuits, data is a level of VDD or GND. Memory elements require a periodic signal (i.e., clock) to synchronize their operations. Data are stored from memory input pins after the clock pulse. This clock pulse triggers memory storage mechanisms. In several memory designs, the procedure to read data is triggered by a clock transition. In this approach, power consumption is minimized by reading data only when it is required. Memory elements can be triggered by the rising (positive) or falling (negative) clock transitions. The type of clock transition trigger determines the sensitive type of memory. Data is read in positive (negative) memory types with rising (falling) clock transition. In some memory elements such as DRAM and SRAM, the storage mechanism requires an enabling signal and clock transition to store data from the input interface. The read procedure of memories may also require an active read signal and clock transition.

2.2.6 Combinational and Sequential Circuits

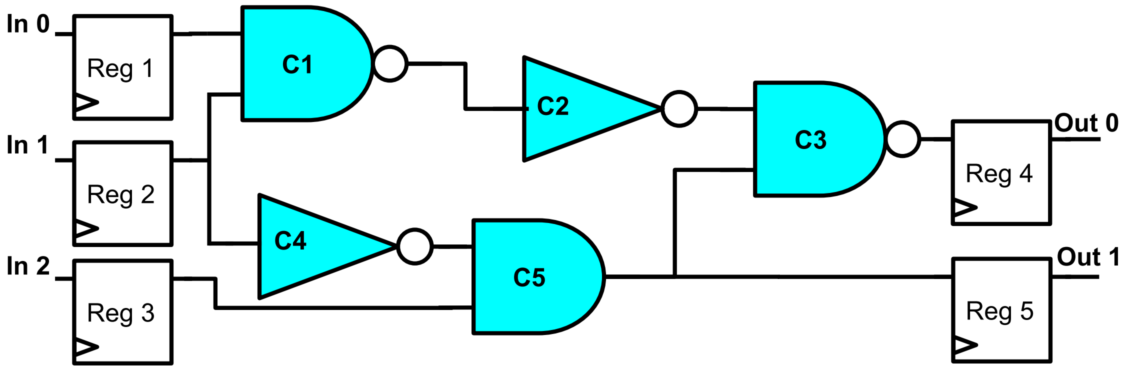
Digital circuits can be classified into *combinational* or *sequential* types. *Combinational circuits* are composed of cells and macroblocks where the output result is only dependent on input data. Therefore, combinational components are independent on the clock signal to trigger internal operations. On the other hand, *sequential circuits* are composed of components that depend on a clock transition to trigger operations. Usually, sequential circuits are composed of registers, memories (e.g., SRAM, DRAM, or cache), pipelines, or FSMs. In Figures 2.6 and 2.7, examples of combinational and sequential circuits are shown, respectively.

Figure 2.6: In combinational circuits, the circuit operation depends only on changing input signals. Combinational circuits do not have memory elements



Source: Author (2019).

Figure 2.7: In sequential circuits, the circuit operation depends on clock transition to launch input data. Sequential circuits have memory elements



Source: Author (2019).

2.2.7 Digital Circuit Designs

Digital circuits fall into *full-custom* or *semi-custom* design types. In *full-custom* design, entire or parts of circuits are designed and implemented manually. Full-custom circuits require a long time to implement. Digital circuits, which are designed with the full-custom methodology, are more error-prone than circuits that are designed with the semi-custom methodology. Errors are hard to detect and fix in the full-custom methodology. On the other hand, circuit delay and power consumption can be further improved. Full-custom methodology fits better in circuits that have severe restrictions on timing and power consumption budgets and an enormous production volume. Usually, microprocessors, Field-Programmable Gate Arrays (FPGAs), Graphics Processing Units (GPUs), and memories are circuit types that have high volume production. These circuit types may have strict requirements in terms of timing and power consumption. *Semi-custom* design style is typically based on standard-cell libraries or circuit arrays (KAHNG et al., 2011). Gate arrays are circuits with predefined logic cells. In gate array methodology, cells are connected later, when circuit requirements are defined.

In FPGAs, logic elements and interconnections are prefabricated. Later, users configure cell logic and connections based on circuit design. FPGAs may be reconfigured numerous times. Structured-Application-Specific Integrated Circuits (ASICs) are similar to FPGAs. However, cell logic is nonconfigurable in Structured-ASICs. Structured-ASICs interconnections are mask-programmed during the fabrication process. In circuit design with standard-cell libraries, circuit logic is synthesized and optimized to predefined logic functions available in standard-cell libraries. In this approach, errors are more straightforward to detect and to fix. However, timing and power consumption improvement compared to full-custom design can be hard to achieve.

2.2.8 Circuit Power and Timing Characterization

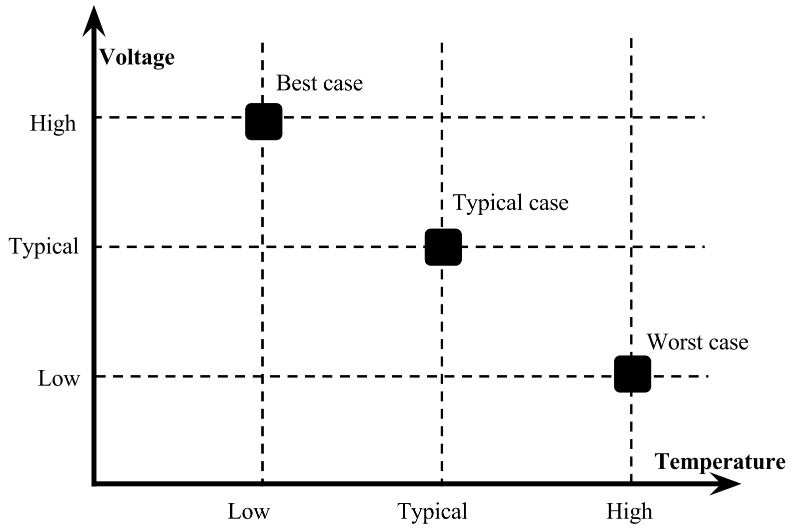
Process, Voltage, and Temperature (PVT) are environmental conditions to characterize standard-cells in a CMOS process node. The maximum performance of standard-cells is achieved in the corner of the best scenario. The maximum performance is obtained with the minimum process variability, the minimum temperature, and the maximum allowed voltage. On the other hand, the minimum performance of standard-cells is related to the worst case scenario. The worst performance is the scenario that is composed of maximum process variability, the maximum temperature, and minimum allowed voltage. Process variability depends on manufacturing parameters of CMOS technology nodes. Digital designers cannot change the manufacturing parameters of CMOS technology nodes.

Standard-cells are typically characterized in best, typical, and worst scenarios for voltage variation and environmental temperature. Standard-cell libraries are characterized to work in a range of environment temperatures from -20 °C to 85 °C. Usually, acceptable voltage drop and voltage bounce is a percentage of the nominal voltage. Standard-cell libraries contain standard-cells which both power consumption and timing are evaluated for several corner cases. A corner case is a pair of voltage and temperature conditions. Timing and power characteristics of standard-cells are measured for discrete sets of input slew transition and output capacitance. For each pair of input slew and output capacitance, electric signals propagate from input pins to output pins in a specific delay. Therefore, the state of the output from standard-cells requires an amount of time to be changed. The output state of standard-cells is switched when the output electric signal is changed from low to high or high to low voltages. Changing the voltage of electric signals in controlling inputs of standard-cells implies the voltage level is changed in standard-cell output. Data of timing and power consumption data from each standard-cell are available in standard-cells libraries for a set of corner cases. Usually, timing and power consumption of standard-cells are characterized for a predefined set of corner cases. In the remaining corner cases, timing and power consumption are interpolated using the closest upper and lower available timing values of input slew and output load capacitance in the standard-cell library tables. In the standard-cell libraries, power component characteristics are split into dynamic and leakage power consumption. In digital circuits, timing and power consumption properties of standard-cells and macroblocks are previously characterized.

Usually, power and timing characteristics of standard-cells are provided using Liberty format. The common Liberty timing models are wire load model, Non Linear Delay Model (NLDP), Composite Current Source (CCS) and Effective Current Source Model (ECSM). The former timing models are more accurate to address sub-nanometric effects of electric signals.

Digital circuits are designed to work correctly in a range of environmental temperatures and voltages. Electric signal propagation is affected by temperature and voltage conditions. Each environmental condition is associated with a corner case, which digital circuits must adequately work. Timing issues are measured in proper corner cases. Usually, in standard-cell libraries, timing and power consumption properties of standard-cells are provided for best, typical and worst cases. Temperature variation depends on circuit external environmental conditions. Usually, the typical temperature is defined as 25° Celsius and the typical voltage is the nominal voltage of the CMOS technology node. Voltage can vary from the nominal voltage of the technology node depending on the internal operations of digital circuits. Acceptable voltage variation is a small percentage of the nominal voltage. Therefore, the voltage in each circuit component must be between the minimum and maximum voltage range. Process variability that is intrinsic to the CMOS technology node is a random parameter related to layout geometries. Circuit parameters are probabilistic instead of static because of the process variability (BORKAR et al., 2003; BORKAR, 2009). In Figure 2.8, the typical set of corner cases to characterize digital circuits are presented.

Figure 2.8: Corner cases to characterize digital circuits. The best case provides the optimal condition to a signal propagation while the worst case provides the inverse condition to a signal propagation



Source: Author (2019).

2.2.9 Digital and Analogical Circuits

Analogical circuits are mainly designed manually with the aid of EDA tools. On the other hand, *digital* circuits are designed relying on EDA tools with an automatized design flow. In digital circuit design, predefined and characterized circuit components are integrated into the die. During circuit implementation, logic functions are optimized. Optimized functions are mapped to predefined circuit components. Mapped components are placed in optimized positions, and component connections are routed. Finally, circuit metrics are evaluated to verify if the circuit layout is ready to be manufactured.

2.3 Static Timing Analysis

In static timing analysis, timing properties of electric signals in digital circuits are estimated using formal methods. Electric signals propagate through metal wire segments and transistors. Metal segments and transistors (semiconductor) have capacitance, resistance, and inductance. Capacitors, resistors, and inductors affect electric signal voltage, current, and delay. In digital circuits, voltage and the delay of electric signals must be reliable. Electric signals must trigger digital components at the correct time.

In the digital design flow, it is only necessary to evaluate the timing properties for wire segments that connect pins of nets. Standard-cells have been characterized in the foundry. STA focus is to compute electric signal properties in wire segments and to interpolate timing characteristics of cells. Timing data of cells are obtained from previously characterized standard-cells. Usually, input slew transition and output capacitance are different from values used to characterize standard-cells. Therefore, the appropriated data timing must be interpolated from timing tables of respective standard-cells.

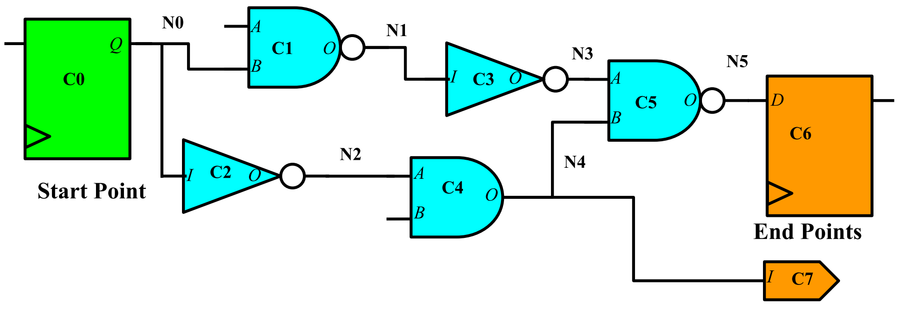
Cell arcs are paths inside cells or blocks in which electric signals are propagated from input to output pins. *Edge arcs* are segments of nets that propagate electric signals from the output pin of driver cells to the input pins of sink cells. *Driver* is the cell which an output pin is the source of the electric signal. *Sink* is a cell in which an input pin receives an electric signal. The same net connects the net driver and the sink pins. Therefore, driver and sink pins have the same electric potential.

In digital circuits, electric signals are propagated from start to end points. Electric signals are launched in start points. These electric signals propagate in wire segments and combinational cells. Therefore, electric signal properties are estimated from the point where the signal is released (start point) to the point where the signal is captured (end point). *Start points* are primary inputs or register outputs. *End points* are primary outputs or register inputs. Pins, ports, and pads, which are external circuit interfaces are *primary inputs and outputs*.

In STA, properties of electric signals are estimated using formal methods. Electric signals propagate simultaneously from start points to end points in parallel paths. Electric signal propagation can be modeled with the aid of a graph. Each start point is the root node of an N-ary tree. Therefore, the number of start points is equal to the number of trees. End points are leaf nodes of trees. Combinational cells are intermediate nodes. Each edge between two nodes represents a connection between two pins of the net. In the tree, each intermediate node has input and output connections. The root has only output connections, and sink nodes have only input connections. In Figure 2.9 and 2.10, a part of a digital circuit and its representative graph are presented, respectively.

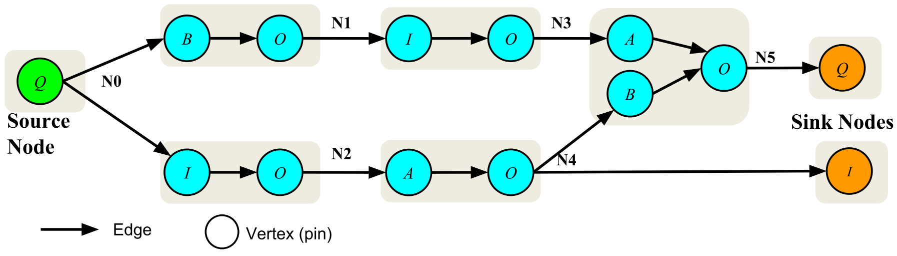
Electric signals do not arrive at the same time in end points. *Arrival Timing (AT)* is the total time that is necessary for electric signals to propagate from a starting point to an ending point. Path AT is associated with the end points. *Required Arrival Timing (RAT)*

Figure 2.9: An example of a part of a digital circuit



Source: Author (2019).

Figure 2.10: Representative STA graph of the circuit in Figure 2.9



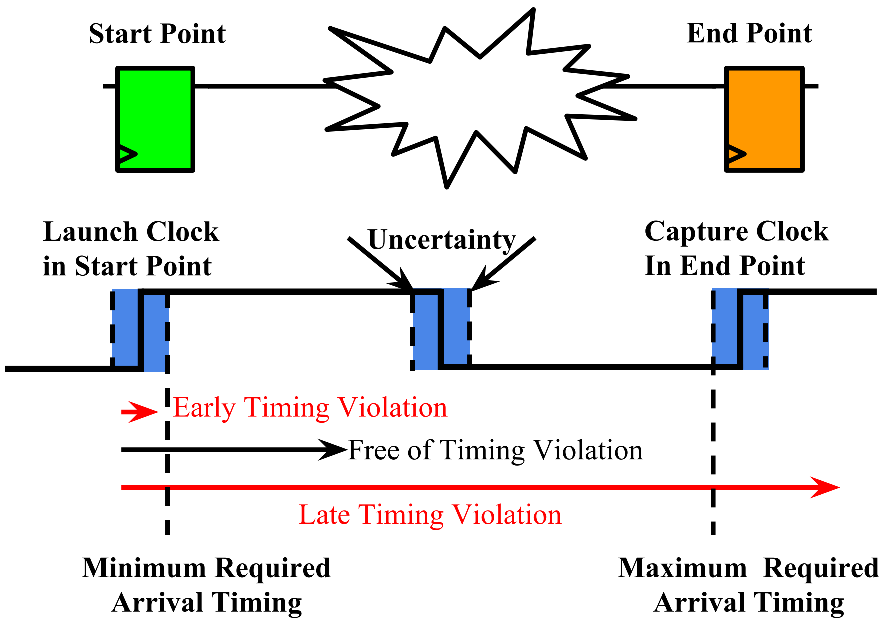
Source: Author (2019).

determines a time window which electric signals must arrive at end points. Usually, the RAT is computed for the best and worst scenarios that give the minimum and maximum RAT window (minRAT and maxRAT), as shown in Figure 2.11. A *path* is a sequence of nodes from the timing graph. The nodes of the paths are connected with net edges. In paths, the first and the last nodes are start and end points, respectively. The timing graph can have several timing paths. A path is free of timing violation if AT is higher than minimum RAT and lower than maximum RAT ($minRAT \leq AT \leq maxRAT$).

Electric signals that arrive at end points earlier than the minimum RAT have timing violations which are denominated *early (hold)* timing violations. The uncertainty in clock transitions is one of the main reasons for early timing violations. The early timing violation occurs due to the following conditions: 1) electric signal delay (AT) is lower than uncertainty timing; 2) the launch clock is triggered at the latest moment of the uncertainty window; 3) the capture clock is triggered at the earliest moment of the uncertainty window. On the other hand, electric signals that arrive at end points later than the maximum RAT have timing violations which are denominated *late (setup)* timing violations. Late timing violations occur mainly due to the cumulative delay to electric signals that propagate through metal wire segments and combinatorial cells. Moreover, these paths with timing violations usually have a considerable number of combinatorial cells and wire segments or longer wire segments.

Launch clock is the trigger to receive and to release data at start points. Capture clock is the trigger to receive and to release data at end points. Both triggers of launch and capture clocks are the same clock edge. The clock transition arrives approximately at the same time at launch and capture points. In Figure 2.11, the launch clock triggers the start point register to capture input data and to release output data. At the same time, the capture clock triggers the end point register to capture input data and to release output data. The data at the input interface of the end point register was captured and released at the start point register in the previous clock cycle. The released data at the start point register must arrive at the input pin of the end point register later than $minRAT$ and earlier than $maxRAT$.

Figure 2.11: Required arrival timing window. A data signal must arrive at the end point after the minimum (early) and before the maximum (late) required arrival timings



Source: Author (2019).

Slack is the difference between AT and RAT, as presented in (2.1). The slack is computed in the same fashion for early and late timing violations. In early timing slack, the mathematical signal (*positive* or *negative*) of slack value is inverted. Inverting the signal of early slack is a convention to indicate timing violations with negative slack values. Negative slack values indicate that electric signals arrive at end points before $minRAT$ or after $maxRAT$. Positive slack values indicate the following condition is valid $minRAT \leq slack \leq maxRAT$. Negative slacks indicate circuit paths with timing violations.

$$slack = \text{RAT} - \text{AT} \tag{2.1}$$

In digital circuits, the most critical timing path is the one that has the lowest negative slack value. The negative slack of the critical path with the lowest value is denominated *Worst Negative Slack (WNS)*. The WNS is computed as presented in (2.2). Total Negative Slack (TNS) is the summation of all negative slacks. WNS and TNS are computed in the same fashion for early and late timing violations. Therefore, digital circuits have early and late WNS and TNS timing metrics. The *TNS* is the summation of negative slack of all critical timing paths, as shown in (2.3).

$$\text{WNS} = min(\forall_{p \in paths}\ slack(p), 0) \tag{2.2}$$

$$\text{TNS} = \sum_{p \in paths} min(0, slack(p)) \tag{2.3}$$

2.4 Electrical Wire Models

Electric signals propagate in wire segments which have resistors, capacitors, and inductors. These components affect electric signal delay, voltage, and current. As a consequence of these effects, circuit performance, power consumption, and reliability can be severely affected. Resistors, capacitors, and inductors are also called parasitic elements in digital circuit design. Propagation characteristics of electric signals in metal wires are estimated based on electrical wire models. Electrical wire models may vary from very simple to very complex models depending on the electric effects to estimate and the required accuracy (RABAEY; CHANDRAKASAN; NIKOLIC, 2003). Available and accurate data of circuit layout and runtime budget may restrict which electrical wire models can be used to estimate wire parasitics. In the early stages of circuit synthesis, the data of the circuit layout is imprecise. Therefore, using precise wire models have no practical effect on obtaining accurate timing characteristics of the circuit. In the later stages of the circuit synthesis, the data of the circuit layout is more accurate. Consequently, precise wire models can be used to estimate wire parasitics with high precision.

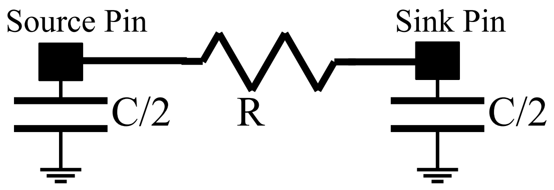
2.4.1 Traditional Electrical Wire Models

The common and the most known wire models are 1) *ideal wire*, 2) *lumped C*, 3) *lumped RC*, 4) π *model*, and 5) *T model*. The *ideal wire model* is free of associated parasitic elements in wire segments. Electric signals propagate instantaneously from the driver to the sink pins without voltage variation or delay caused by parasitic elements on wire segments. In the *lumped C* model, wire capacitance is modeled as a capacitor attached to the output pin of the driver cell. Wire resistance has no significant effect on the delay of electric signals in wire segments. Therefore, the wire capacitance is the dominant parasitic element compared to wire resistance. In the lumped C wire model, the delay of electric signals is mainly due to the load time of wire capacitance (RABAEY; CHANDRAKASAN; NIKOLIC, 2003).

Lumped RC model addresses wire capacitance and resistance as a single resistor (R) and a single capacitor (C). In this wire model, the electric signal delay is inaccurate and pessimistic for long wires.

In the π *wire model*, half of the wire capacitance is a capacitor associated with the driver pin. The remaining half capacitance is a capacitor associated with the sink pin. A resistor is connected to the source and the sink pins. In Figure 2.12, π wire model is presented.

Figure 2.12: π wire model. Electric properties of wire segments are modeled with two capacitors that are connected by resistor

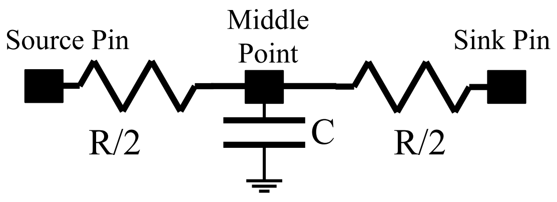


Source: Author (2019).

In the *T wire model*, the total wire capacitance is modeled by a capacitor connected to the middle node of the wire. The wire resistance is split into two resistors. The first resistor is connected to the driver pin and to the middle node. The second resistor is connected to the middle node and to the sink pin. In Figure 2.13, *T wire model* is shown.

Elmore (ELMORE, 1948) is a simple method to estimate delay in a structured RC wire tree. In the Elmore model, only capacitance and resistance is considered for computing the electric signal delay. Estimated electric signal delay with the Elmore model is imprecise and fast to compute. However, for sub-nanometric CMOS technologies, elec-

Figure 2.13: T wire model. Electric properties of wire segments with two resistors and one capacitor. One terminal of each resistor is connected to the capacitor



Source: Author (2019).

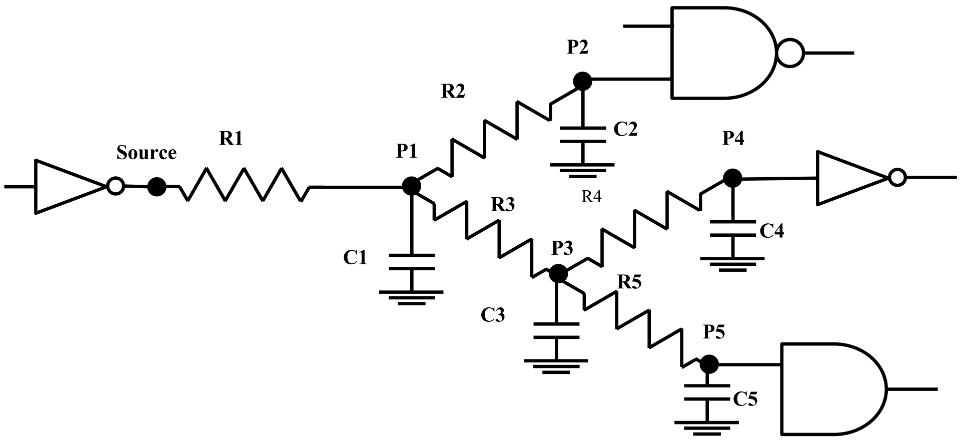
tric effects in propagation properties of electric signals are ignored. The Elmore equation to compute delay of electric signals is presented in (2.4).

$$\tau_{Di} = \sum_{k=1}^{N} C_k \sum_{j=1}^{k} R_j \tag{2.4}$$

where, τ_{Di} is the delay in the node i of the structured RC tree network. N is the number of nodes in the structured RC tree network. R_j is the resistance in the node j ($1 \leq j \leq k$). C_k is the capacitance associated with node k.

In Figure 2.14, an example of a structured RC tree network is presented. Delay of the electric signal from source to P5 is estimated using (2.4) as follows: $\tau_{D\_p5} = R_1(C_1 + C_2) + (R_1 + R3)(C_3 + C_4) + (R_1 + R_3 + R_5)C_5$.

Figure 2.14: Example of a structured RC tree network. Electric signal delay is computed from source to P5 points with Elmore model



Source: Author (2019).

In the aforementioned simplified wire models, inductance is neglected. It is assumed that inductance has no significant impact on the delay properties of electric signals.

In placement, especially in nodes older than 14nm, structured RC tree network, Elmore and π wire model are considered adequate methods to estimate the delay of electric signals. These methods are relatively precise, fast to compute, and can be used in STA at the placement step to evaluate circuit timing. Frequently, the length of routed wires are estimated using a fast global router or by building rectilinear Steiner trees (e.g., Flute (CHU; WONG, 2008)). However, the fast estimation of routed wires can have an enormous gap between the wire length of estimated and routed nets. Besides, the metal layer assignment can significantly change the delay properties of electric signals.

Computing precise wire capacitance, resistance, and delay can become very sophisticated in digital circuits. Notably, in digital circuits that have a large number of these elements require a significant runtime to estimate the signal delay. There are several methods to estimate the properties of electric signals that are more precise than the wire models, as mentioned above, and Elmore. Some of these more precise wire models are 1) PRIMA (ODABASIOGLU; CELIK; PILEGGI, 1998), 2) (CHUNG et al., 1992), 3) (PILLAGE; ROHRER, 1990), and 4) SPICE (NAGEL; PEDERSON, 1973). Some of these methods also address the effects of electric signals in nanometric CMOS technologies. However, accurate methods to estimate properties of electric signals require significantly more computational resources. There is a trade-off between the accuracy of the timing properties of electric signals and the required computational resources.

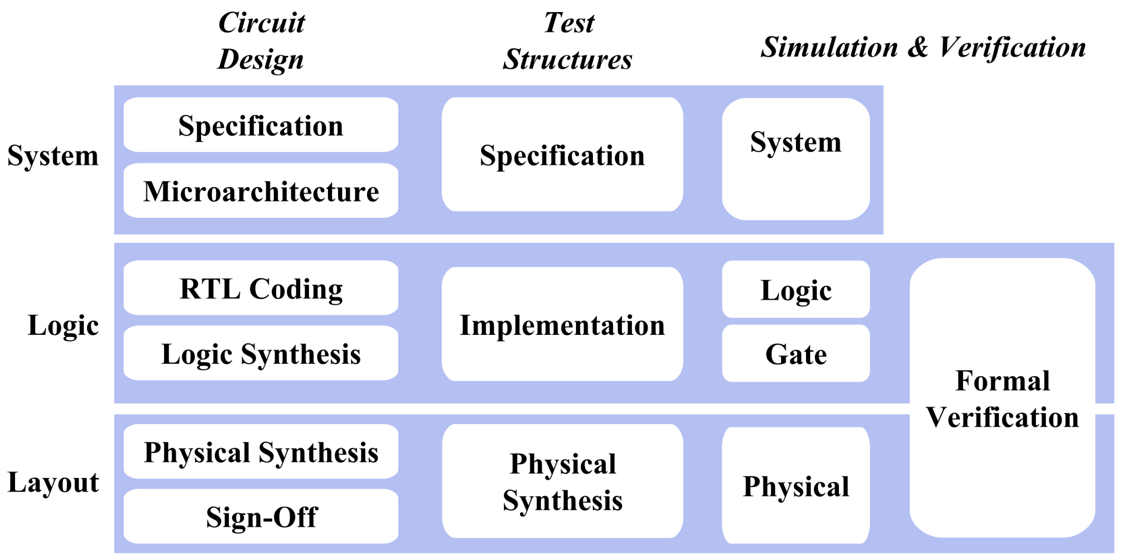
2.5 Digital Design Flow

Digital design flow is a top-down sequence of procedures to elaborate and to synthesize digital circuits. Circuit features are described in natural language and processed following several synthesis and optimization steps of the digital design flow. The result of digital circuit synthesis and optimization is a circuit layout which is ready to tape-out. Data of circuit layout becomes more precise after each synthesis step in the digital design flow. New data regarding circuit geometry and electric characteristics are included in the circuit design during the synthesis steps.

2.5.1 Design flow

Digital design flow is composed of three main flows which are operated in parallel. These design flows are 1) *circuit design*, 2) *test structures* and 3) *simulation & formal verification*, as shown in Figure 2.15. Each parallel flow is composed of several sequential steps. Digital design flow can be split into three levels of abstractions. These abstraction levels are 1) *System*, 2) *Logic* and 3) *Layout*. At the system level, the circuit design is defined and characterized in natural language and diagram of blocks. At the logic level, the circuit design is defined and characterized by optimized-Boolean functions. At the layout level, the circuit layout is defined with regular geometries. These geometries are representations of rules from the manufacturing process.

Figure 2.15: Digital design flow comprises of three main parallel flows that are split into several steps. The circuit implementation starts with requirement documents. After several synthesis, optimization, verification, simulation steps, the circuit layout is ready to be manufactured



Source: Author (2019).

In *circuit design* flow, several steps are performed to optimize and to synthesize the digital circuit. The result is a circuit layout ready to be manufactured. At the *system level*, circuit requirements, features, behaviors, and restrictions are described in specification documents using natural language. The microarchitecture step is to organize circuit features in a diagram of blocks. In each block, specific circuit operations are performed. Communication protocols and bus connections among blocks are also designed at the microarchitecture level. In *logic level*, circuit features of each circuit block are coded in RTL using a Hardware-Description Language (HDL). At the logic synthesis step, Boolean functions are optimized. These functions have been coded in RTL to describe the circuit behavior. Optimized logic functions are mapped to standard-cells from a library cell of a particular CMOS technology node. At the *layout level*, the floorplan of the circuit is determined, cells are placed in optimized positions, and nets are routed. Finally, in the sign-off step, circuit metrics are estimated, and restrictions are verified if they adhere to the circuit requirements. Circuit metrics are measured using high precision methods and algorithms. The results of the synthesis and the optimization steps are a circuit layout that is ready for the manufacturing process.

2.5.2 Test Structure Flow

In test structure flow, extra logic for testing is added to circuit design and layout. Test structure facilitates checking the correctness of circuit operations after the fabrication process. Manufacturing problems, such as short circuit or open connections, are detected with the aid of the test structures. Digital circuits are tested by applying a set of stimulus vectors in circuit inputs and evaluating if output data are the expected values. Test structures also aid to expose internal results to the circuit output interface. The typical test structures are scan chain and Built-In Self-Test (BIST).

In scan chain, registers store combinational logic results or data of previous registers. An external signal controls and synchronizes which data are stored in registers. In this approach, internal logic results are exposed to the output interface and compared to expected results.

BIST is a protocol to perform circuit autotest. In BIST, the circuit generates internal stimulus and evaluates expected results to detect errors. BIST can be automatically performed when the circuit is turned on.

2.5.3 Formal Verification and Simulation

Formal verification and simulation flow are two stages to verify and simulate circuit netlist. In the formal verification stage, circuit netlists are verified with formal methods to detect errors. Pre-optimized and post-optimized netlists are verified using formal methods if they are logically equivalent. In the simulation stage, circuit behavior is simulated using EDA simulation software. Simulation is frequently used to estimate dynamic power consumption, to detect power hot-spot areas, to address regions with voltage drop or voltage bounce. Circuits are simulated in parallel with flows for the optimization, and test structures. Simulation is a process to evaluate circuit behavior and characteristics using a set of distinct input vectors. A set of stimulus is applied in circuit inputs, and the outputs are evaluated. The circuit is also simulated to estimate circuit metrics and to detect logic errors. Errors in optimization and synthesis algorithms can cause logic errors. Logic errors can also be caused by digital designers who have misunderstood circuit restrictions and features in design specification documents. These errors may modify circuit functionalities. Therefore, formal verification and simulation are stages in the flow to detect unwanted changes in circuit logic.

2.5.4 Physical Synthesis Flow

The circuit layout is optimized in the physical synthesis flow. Physical synthesis can be divided in a flow composed of four stages: 1) *Floorplan*, 2) *Placement*, 3) *CTS*, and 4) *Routing*, as presented in Figure 2.16. During physical synthesis flow, circuit metrics are estimated and evaluated in parallel.

Layout evaluation flow is composed of 1) *parasitic extraction*, 2) *timing analysis*, 3) *signal integrity*, 4) *power analysis*, 5) *routing congestion*, and 6) *design rules check*. The initial step of physical synthesis flow is to determine the circuit floorplan. During floorplanning, the following procedures are performed.

1. Circuit shape is defined.
2. Power rings are designed and routed.
3. IO pins' or pads' positions are assigned.
4. Usually, macroblocks are placed.

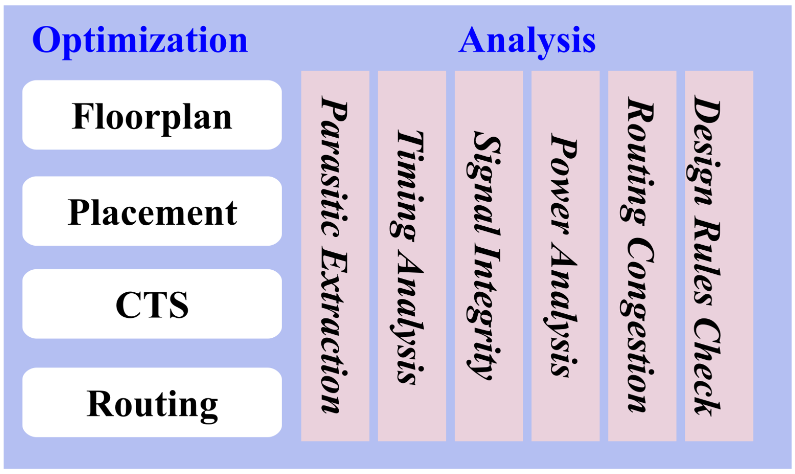
During the placement step, optimized cell positions are computed to achieve circuit objectives subject to certain restrictions. Typical placement objective is to optimize the total wire length. Placement relevant restrictions are design rules, maximum area density utilization, power consumption, and timing violations.

Clock tree synthesis is the next step in the physical synthesis flow. Clock tree has unique objectives and constraints compared to the routing process of interconnections of data signals.

In the routing step, specialized algorithms route only interconnections of data signals. After the routing step, the circuit layout is finished. The circuit layout is ready to estimate and to evaluate precise circuit metrics. Circuit requirements and objectives are verified if they have been achieved.

Parallel to physical synthesis flow, specialized algorithms are used to evaluate design metrics such as timing propagation, power consumption, signal integrity, and voltage drop. In some cases, when a physical synthesis step has many intermediate steps, circuit metrics may be evaluated after the execution of intermediate steps. These metrics aid synthesis and optimization algorithms to focus the layout improvement in regions with bottlenecks. Evaluated metrics can also be used to estimate rough layout results of the following physical synthesis optimization algorithms.

Figure 2.16: The physical synthesis flow that is part of the digital design flow is split into several sequential optimization steps and parallel stages to estimate design metrics



Source: Author (2019).

2.6 Summary

In this chapter, concepts and definitions associated with digital circuit design have been introduced. The main circuit conventions have been presented to avoid misunderstandings regarding the semantics of circuit terms. Circuit timing metrics and evaluation methods have been introduced. Timing analysis should be fast in order to be efficiently computed during the synthesis process. Reliable timing information aims to improve the quality of the circuit layout. The timing data is used to guide synthesis and optimization algorithms to reduce bottlenecks in the synthesized circuit solutions. Reliable models are essential for estimating circuit timing and power metrics. Available wire models have a trade-off between required computing resources and the accuracy to estimate properties of electric signals. During placement, the Elmore delay model is widely used due to fast delay estimation and its relative accuracy.

Digital circuits are designed, optimized, simulated, and verified using EDA algorithms and tools intensively. These tools and algorithms are organized in a digital design flow. This design flow is composed of several parallel and sequential synthesis, optimization, simulation, and verification steps. In the first stage of the digital design flow, circuit requirement documents are elaborated. Based on requirement documents, circuit microarchitecture is designed. Circuit requirements and microarchitecture are coded in RTL with a HDL in a netlist. The logic netlist is optimized and mapped to standard-cells of a library cell from a CMOS technology node. Circuit cells are distributed in the predefined core die. Heuristics and formal optimization techniques are fundamental to determine optimized cells' positions. Finally, connections of pins are routed, and circuit metrics are precisely estimated. Circuit requirements and restrictions are evaluated if they have been achieved. The result of synthesis, optimization, verification, and simulation procedures in the digital design flow is a circuit layout ready for the CMOS manufacturing process.

In digital design, test structure flow and simulation & formal verification flow are sequences of steps to implement test structures, to simulate and to verify the circuit netlist. Test structures are specified, designed, and synthesized to detect circuit issues that can be caused during the manufacturing process. Digital circuits are simulated and formally verified. Digital circuits are simulated to estimate circuit metrics and to detect logic errors. In formal verification, the logical equivalence of circuit netlists is verified to detect if synthesis and optimization algorithms have caused logic errors.

3 ELECTRONIC DESIGN AUTOMATION

3.1 Introduction

In the 1960s, the first EDA tools started to appear to assist circuit designers (KAHNG et al., 2011). Ever since, CMOS fabrication technology and EDA tools have significantly evolved. The process of designing, implementing, and verifying integrated circuits have been significantly automated with EDA tools. The evolution of CMOS fabrication technology imposes complex design rules to manufacture modern digital circuits. Especially in sub-nanometric technology nodes, the number of design rules has grown exponentially. Circuit optimization techniques require extensive use of heuristics. The formulation of circuit optimization problems is very complicated. Several optimization techniques depend on formal models to optimize circuit objectives. Moreover, circuit optimization is subject to several circuit and design restrictions.

The first EDA tools were placement algorithms. These placement algorithms were developed to automatize the distribution of a small number of blocks on circuit boards. More advanced EDA tools to visualize and implement circuit layout have been developed. In the 1970s, EDA tools were developed to aid circuit design synthesis and optimization (KAHNG et al., 2011). In the 1980s, independent software companies started developing EDA tools. Nowadays, EDA tools have become very complicated. Modern EDA tools have to handle a large number of restrictions in digital circuits composed of thousands to millions of cells. Digital circuit requirements, features, and restrictions have also increased significantly. In digital circuits, obtaining feasible optimized circuit solutions is a challenging procedure. Digital circuits are synthesized, optimized, verified, and simulated with the heavy use of EDA tools. These EDA tools are integrated into digital design flows to implement digital circuits.

By the middle of the 2000s, analytical placement algorithms had significantly improved. These placement algorithms provided similar placement solutions compared to partitioning-based and stochastic techniques. However, the analytical placement algorithms of the mid-2000s required less runtime to provide similar placement solutions compared to other placement techniques. Partitioning-based and stochastic techniques became hard to scale because of the rapid growth of the number of cells and restrictions in modern digital circuits.

This chapter is organized as follows: In Section 3.2, algorithm definitions are introduced. In Section 3.3, graph definitions are shown. In Section 3.4, optimization techniques including network flow and branch and cut are shortly presented. In Section 3.5, models to decompose hyperedge nets into nets with only two-points connections are shown. In Section 3.6, metrics are introduced to evaluate and to estimate layout optimization in physical synthesis flow. In Section 3.7, construction of the graph of grid bins for cell spreading algorithms that are based on network flow technique is presented. In Section 3.8, formulation of EDA algorithms in physical synthesis flow is highlighted. Finally, the chapter summary is given in Section 3.9.

3.2 Algorithm Definitions

An *algorithm* is a finite sequence of instructions well-defined in a formal language. Algorithm instructions are executed in a finite amount of time and space. Algorithms may or may not receive input data. Algorithms must produce at least one output data. An algorithm must have the following properties (HOROWITZ; SAHNI; RAJASCKARAN, 1996):

- *Finiteness*: An algorithm must terminate after a finite number of operations.
- *Definiteness*: Each step of an algorithm must be precise, rigorously, and unambiguously specified.
- *Input*: An algorithm may have external data input.
- *Output*: An algorithm must provide at least one data output.
- *Effectiveness*: Algorithm operations must be sufficiently basic so that algorithm instructions may be precisely executed in a finite amount of time.

Algorithms may be classified into 1) *deterministics*, or 2) *nondeterministics*. *Deterministic algorithms* always produce the same output passing through the same states for the same input. *Nondeterministic algorithms* produce different output solutions which depend on external input data (e.g., a random value, or user input data) passing through different states for the same input.

Algorithms are also classified as 1) *iterative* or 2) *direct*. *Iterative algorithms* successively provide an approximate solution in each iteration. The new solution is closer to the optimum solution compared to the previous solution. *Direct* algorithms try to provide the problem solution by a finite sequence of iterations.

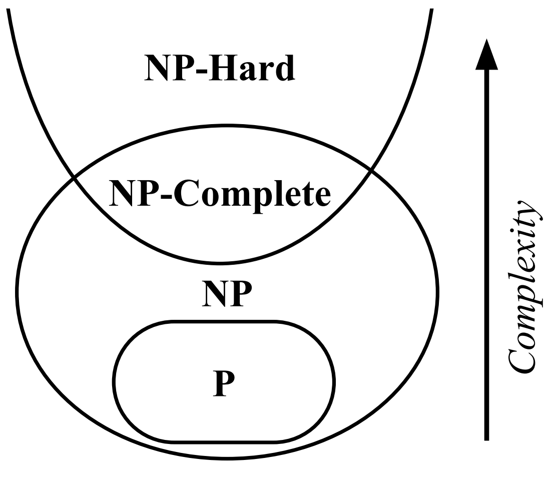
An algorithm requires a certain amount of computational resources (e.g., memory space, or timing) to provide a solution. Required computational resources depends on problem type. Problem types are related to class and size of input data of computational problems. Computational problems are classified into 1) *Polynomial Time (P)* and 2) *Nondeterministic Polynomial Time (NP)* problem classes (HOROWITZ; SAHNI; RAJASCKARAN, 1996). Formal definitions of class P and NP problems are presented in Definitions 3.1, and 3.2, respectively.

Definition 3.1 *P is the set of all decision problems solvable by deterministic algorithms in polynomial time.*

Definition 3.2 *NP is the set of all decision problems solvable by nondeterministic algorithms in polynomial time.*

It is not known if deterministic polynomial time algorithms can provide the optimum solution of NP problems. There is no formal proof the P and NP classes are equal. If P and NP classes are equal, then NP problems have deterministic polynomial time algorithms which provide the optimum solution. Nowadays, optimized solutions for several NP problems are obtained in polynomial time with intensive use of heuristics in deterministic algorithms. Definitions of NP-hard and NP-complete problem classes are not given in this section. These class problems have not known deterministic polynomial time algorithms to provide the optimum solution. In Figure 3.1, the diagram of intersection among classes P, NP, NP-complete and NP-hard problems is presented.

Figure 3.1: Diagram of intersection among classes P, NP, NP-complete and NP-hard problems



Source: Author (2019).

Algorithm runtime can be roughly estimated with *asymptotic analysis* (*big-O notation*) by assuming a large data input (SIPSER, 1996). In Definition 3.3, formal definition of asymptotic analysis is presented (SIPSER, 1996).

Definition 3.3 *Let f and g be two functions $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$ that $f(n) = O(g(n))$ if positive integers c and n_0 exist for every integer $n \geq n_0$ in $f(n) \leq cg(n)$.*

The function $g(n)$ is *asymptotic upper bound* of function $f(n)$. Asymptotic functions are given as the following $f(n) = O(g(n))$. Constant factors are suppressed to emphasize only the asymptotic upper bound.

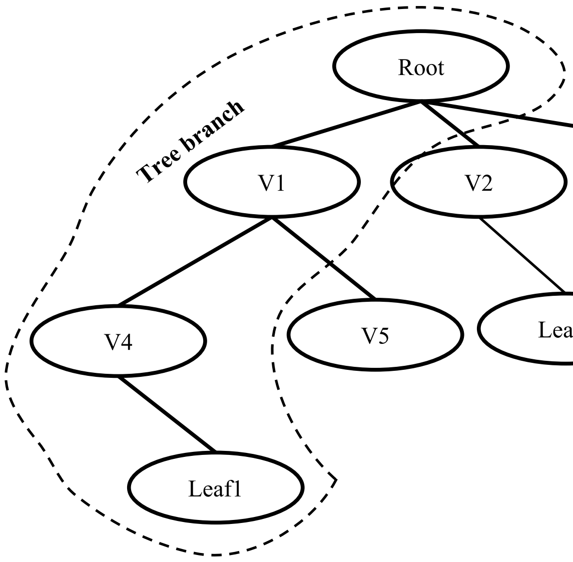
3.3 Graph Definitions

A *graph* G is a tuple (V, E, R) (WEST, 2001) where $V \in G$ is a nonempty set of *vertices* or *nodes*, $E \in G$ is a set of *edges*, and $R \in G$ is a set *relations* that connects vertices to edges. Vertices $v_i, v_j \in V$ are connected through edge $e \in E$ in graph G. In *hypergraph*, vertices $v_1, \ldots, v_n \in V$ ($n > 2$) are connected though a *hyperedge* $e \in E$. A *multigraph* has multiple edges connecting the same pair of vertices. A *directed graph* or *digraph* is a graph where edges have direct connections, instead of non-direct connections.

A *walk* in a graph G is a finite non-null sequence of vertices and edges whose terms are alternately vertices and edges ($v_0 e_1 v_1 e_2 v_2 \ldots e_n v_n \in G$). A *path* in a graph G is a sequence of adjacent vertices ($v_0 \ldots v_n \in G$), where each vertex is connected to the subsequent vertex by an edge with only distinct vertices (BONDY, 1976). A *cycle* is a path where adjacent vertices are connected in a circle. In cycles, the last and the first vertices are connected ($v_0 \ldots v_n v_0 \in G$) with an edge $e \in E$. A graph composed of several disjoint trees and free of cycles is a *forest* (WEST, 2001). A tree is a graph free of cycles with which two vertices are connected by only one path. In a tree, vertices are connected to parent and children vertices, except root and leaf vertices. Root vertex has connections only to children vertices. Leaf vertex has only a connection to the parent vertex. A tree branch is a path with which the first and the last vertices are root and leaf, respectively. In Figure 3.2, an example of a graph tree is presented.

Graphs may be represented with adjacency lists, adjacency matrices or incidence matrices. In the adjacency list, each vertex stores a list of neighboring vertices. In the adjacency matrix, graphs are represented in a two-dimensional matrix where rows and columns represent vertices. Each matrix position p(i,j) ($0 \leq i \leq numRows$, $0 \leq j \leq numColumns$) represents a possible edge. Marked positions indicate edge connections between two vertices. In the incidence matrix, a two-dimensional matrix represents vertices and edges in rows and columns, respectively. A marked matrix position p(i,j)

Figure 3.2: An example of a graph tree



Source: Author (2019).

$(0 \leq i \leq numRows, 0 \leq j \leq numColumns)$ indicates the vertex in row i is connected to the edge in column j.

In EDA, *netlist* describes net connections among pins of circuit cells, modules, macroblocks, IO pins and pads. The circuit netlist can be modeled as a graph $G = (V, E)$. The set of vertices V represent pins. Pins are the input and output interfaces of circuits, cells, macroblocks, and pads. The set of edges E represent circuit nets. Vertices are connected with edges (nets) or hyperedges (hypernets). Each net $e_i \in E$ is electrically connected to a subset of circuit pins of V (ALPERT; MEHTA; SAPATNEKAR, 2008).

3.4 Optimization Algorithms

In 3.1, a standard formulation of an optimization problem is presented.

$$\begin{aligned} &Minimize\ f(x) \\ &Subject\ to\ g_i(x) \leq 0,\ i = 1, \ldots, m \\ &\qquad\qquad h_j(x) = 0,\ j = 1, \ldots, p \end{aligned} \tag{3.1}$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective function to be optimized over a n-variable vector x. $g_i(x)$ are inequality restrictions. $h_j(x)$ are equality restrictions. $m \geq 0$ and $p \geq 0$. The optimization problem is unconstrained if $m = 0$ and $p = 0$.

In an optimization problem, the objective is to find the best solution among feasible solutions. Optimization problems can be classified into *discrete* or *continuous* types. In discrete optimization problems, feasible solutions are searched in discrete sets (integers, graphs, and permutations). In continuous optimization problems, feasible solutions are searched with continuous variables (e.g., $\mathbb{R}$).

Optimization problems have an enormous variety of techniques to compute feasible solutions. In this section, network flow and branch and cut techniques are discussed. These techniques are relevant to the proposed algorithms in this thesis. Based on these techniques, optimized circuit solutions are provided with the proposed algorithms.

3.4.1 Network Flow

Network flow (transportation network or flow network) problem is a class of combinatorial optimization problems. Network flow can be used to model elements which can be transported through edges and vertices of graphs. A network flow is a graph where edges have transportation capacity and can receive and provide flow. The flow amount must not exceed the capacity of the edges. Network flow $N(G(V, E), s, t, c)$ is defined as follows (EVEN, 2011):

- A finite digraph $G(V, E)$ where V is the set of vertice and E is the set of edges.
- Source ($s \in V$) and sink ($t \in V$) vertices.
- Edge capacity function $c(e) : E \rightarrow \mathbb{R}^+$.

The result is a flow which inflow is equal to outflow in vertices, except for source and sink vertices.

The set of edges that enter and emanate from a vertex $v \in V$ is denoted as $\alpha(v)$ and $\beta(v)$, respectively (EVEN, 2011). The flow function $f : E \rightarrow \mathbb{R}$ is the assignment of a real number $f(e)$ which indicates the flow associated with an edge. The flow $f(e)$ must be higher than or equal to 0 and lower than or equal to the edge capacity ($e \in E, 0 \leq f(e) \leq c(e)$). $Inflow$ must be equal to $Outflow$ in each vertex as introduced in 3.2. $Inflow$ is the summation of the flow for input edges in each vertex. $Outflow$ is the total flow of output edges in each vertex.

$$\sum_{e \in \alpha(v)} f(e) = \sum_{e \in \beta(v)} f(e) \tag{3.2}$$

In the network flow literature, the common algorithms to obtain maximum flow are 1)

Ford-Fulkerson (FORD; FULKERSON, 1987), 2) (DINIC, 1970), 3) Edmonds-Karp (EDMONDS; KARP, 1972), 4) (MALHOTRA; KUMAR; MAHESHWARI, 1978), and 5) (ORLIN, 2013).

The network flow problem has several minimum cost flow models such as, 1) *shortest path*, 2) *maximum flow problem*, 3) *assignment problem*, 4) *transportation problem*, 5) *convex cost flow*, 6) *generalized flow*, and 7) *multicommodity flow* (AHUJA; MAGNANTI; ORLIN, 1993). In *shortest path problem*, the objective is to find the path with the minimum cost or minimum length from *source vertex s* to *sink vertex t*. Each edge $e(i, j) \in G$ has the cost or length c_{ij} associated with it. In the *maximum flow problem*, the objective is to find a solution in which the maximum flow from a *source vertex s* can be sent to a *sink vertex t*. Each edge $e(i, j) \in G$ has the maximum flow u_{ij} associated with it. In the *assignment problem*, the minimum cost flow problem consists of two equal size sets' S_1 and S_2 . Pairs ($P \in S_1 \times S_2$) indicate possible assignment. Cost (c_{ij}) is associated with the pair P_{ij}. The objective is to obtain the pair with the minimum cost where objects in $S_1 \in P$ have respective objects in $S_2 \in P$. In the *transportation problem*, the set of vertices N is partitioned in two subsets $N_1 \in N$ and $N_2 \in N$. N_1 is the set of supply vertices. N_2 is the set of demand vertices. Supply and demand vertices can provide and receive flows, respectively. Each edge between vertices N_i and N_j has $N_i \in N_1$ and $N_j \in N_2$. In the *convex cost flow*, the cost of each edge has a linear convex cost function. In *generalized flow*, flow can be consumed or generated in intermediate edges. A flow can enter in an edge, and part of this flow can be consumed by this edge. On the other hand, additional flow can be added to the flow that has entered in the edge. In *multicommodity flow*, several commodities can share the same edge. Their characteristics, origins, or destinations can be different for different commodity types. Commodity constraints may also be different.

3.4.2 Branch and Cut

Linear Programming (LP) is a formal method to optimize objectives of constrained linear functions. In LP, objectives and constraints are linear functions. In Integer Linear Programming (ILP) problems, some or all variables are integer values. The objective and constraint functions are linear.

Branch and bound (LAND; DOIG, 1960) is an algorithm that systematically enumerates candidate solutions. This algorithm explores candidate solutions in branches of an n-ary tree. In each branch, the candidate solution is compared to the lower and upper bounds of the estimated optimal solutions. The algorithm opens branches only if a branch candidate solution is better than the lower and the upper bounds. Otherwise, the branch solution is discarded.

In cutting plane (GOMORY, 1958), new constraints are added to the linear problem formulation. The objective is to find feasible solutions with integer values. These extra constraints are called *cut*. Cuts related to fractional solutions must satisfy the following criteria: 1) integer solutions are feasible for cut constraints and 2) noninteger solutions are not feasible for cut constraints.

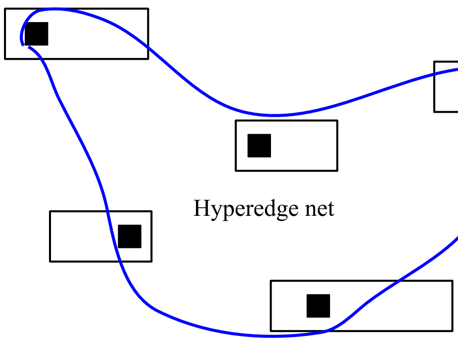
Branch and cut (PADBERG; RINALDI, 1991) is a method to solve ILP problems when some or all unknown variables are integers. The branch and cut algorithm consists of the branch and bound with cutting plane methods to generate candidate solutions. N-ary tree branches are pruned if candidate solutions are lower or upper than existing lower and upper bound solutions. Branch strategies can be classified into 1) *most infeasible branching*, 2) *pseudo cost branching*, 3) *strong branching*, and 4) *full strong branching* (ACHTERBERG; KOCH; MARTIN, 2005). In most infeasible branching, the variable with fractional part closest to 0.5 is chosen to open. In pseudo cost branching, the change in the objective function in each variable is tracked for the variables which were previously chosen to branch on. The variable which is predicted to have the most change in past changes is chosen to branch on. In strong branching, candidate variables are tested to obtain the one which gives the best improvement before branch procedure. In full strong branching, all candidate variables are tested before branch procedure.

3.5 Models to Decompose Hyperedge Nets

In global placement, quadratic placement algorithms can compute optimized placement only in a circuit with nets that have only two pin connections. Therefore, hyperedge nets must be decomposed into a set of edges that connect only two pins. Hyperedge nets are decomposed using a net model that evenly addresses some net characteristics, such as wire length. Common net models to decompose hyperedge nets are 1) *clique*, 2) *star*, 3) *hybrid*, and 4) *Bound to Bound (B2B)*. In Figure 3.3, a hyperedge net is presented. This hyperedge has connected five cells. The hyperedge net, as mentioned above, is decomposed with clique, star, hybrid, and B2B net models.

In the clique net model, hyperedge nets that connect k-pins are decomposed into

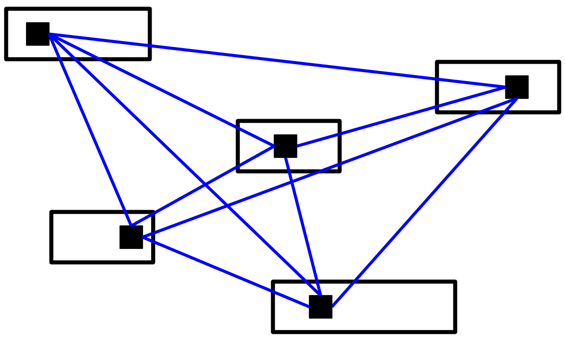
Figure 3.3: Hyperedge nets must be decomposed into a set of pair connections between only two points using a net model. In this example, the net with five pins is decomposed into clique, star, hybrid and B2B net models



Source: Author (2019).

$\frac{k(k-1)}{2}$ pair connections. In Figure 3.4, a five-pins hyperedge decomposed with the clique net model net is presented.

Figure 3.4: All net pins are connected to each other when hyperedge nets are decomposed with the clique net model



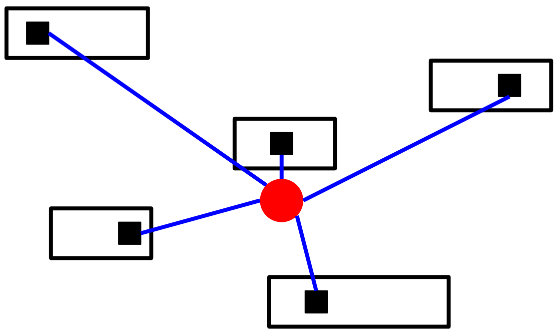
Source: Author (2019).

In the star net model, each hyperedge net has a central star node to connect all pins of the net via pair connections. In each pair connection, each pin of the net is connected to the star node. Nets with k-pins have k pair connections to star node. In Figure 3.5, a five-pins hyperedge net decomposition with the star net model is presented.

In Figure 3.6, the hybrid net model is presented. Hyperedge nets that have up to k pins are decomposed using the clique net model, and the remaining hyperedge nets are decomposed using the star net model. Usually, k is limited to three pins.

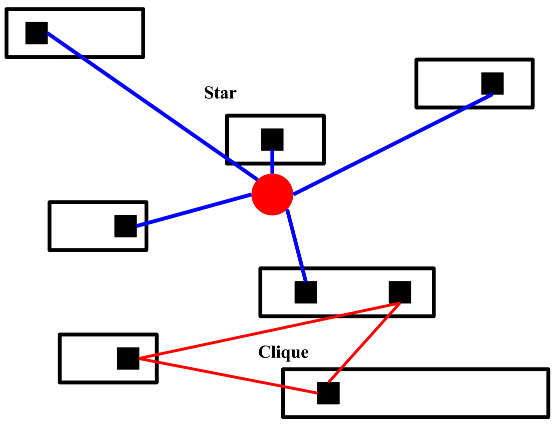
In the B2B net model (SPINDLER; SCHLICHTMANN; JOHANNES, 2008b), hyperedge nets are decomposed into pair connections inside of the net bound box. In B2B, pins of hyperedge nets are sorted by their positions in independent vectors of pins for abscissa and ordinate axis. These pins are classified into outer and inner pins for abscissa and ordinate axis. Outer pins establish boundaries of the net-bound box. Therefore,

Figure 3.5: All net pins of hyperedge nets are connected to a central point in the star net model



Source: Author (2019).

Figure 3.6: In the hybrid net model, hyperedge nets that have up to 3 pins are decomposed using the clique net model. Remaining hyperedge nets are decomposed using the star net model



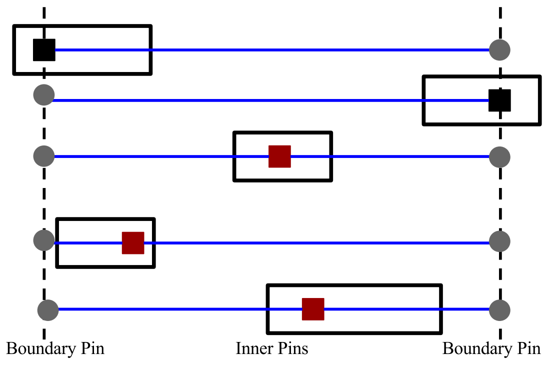
Source: Author (2019).

outer pins have the lower and upper positions of the bound box in each Cartesian axis. The remaining pins are classified as inner pins. In each Cartesian axis, all net pins are connected only to the lower and upper boundaries of the net bound-box, as shown in Figure 3.7. In the B2B net model, boundaries for Cartesian abscissa and ordinate are independently defined.

In analytical placement formulation, weights are associated with nets. Weight values indicate the relevance of nets regarding objectives to optimize or restrictions to be attended. Net weight indirectly and evenly prioritizes some circuit characteristics (e.g., wire length, or timing delay) to be optimized using numerical methods. Global placement common net weights to linearize quadratic wire length are introduced in (3.3) (VYGEN, 1997) and (3.4) (KLEINHANS et al., 1991; EISENMANN; JOHANNES, 1998).

$$net(w) = \frac{1}{k-1} \tag{3.3}$$

Figure 3.7: In the B2B net model, net pins are classified into inner and outer pins. All pins of each net are connected to boundaries of net bound box. Two independent B2B net models are built for abscissa and ordinate axis of each net



Source: Author (2019).

$$net(w) = \frac{2}{k} \tag{3.4}$$

where k is the number of net pins.

In the B2B model, net weight can be computed as presented in (3.5).

$$W^{b2b}_{x,pq} = \begin{cases} 0, & \text{if } p \text{ and } q \text{ are inner pins} \\ \frac{2}{P-1}\frac{1}{|x^{pin}_p - x^{pin}_q|}, & \text{else} \end{cases} \tag{3.5}$$

where p and q are two pins of the net. P is the number of net pins. x_p and x_q are positions of pins p and q in abscissa, respectively. In ordinate, net weight is computed in the same fashion.

3.6 Circuit Evaluation Metrics

In digital circuit optimization, circuit metrics are essential to evaluate the quality of the optimized circuit solution. Moreover, circuit metrics are used to guide the optimization and synthesis algorithms to prioritize optimization of specific characteristics. Circuit metrics may also be used to guide optimization algorithms to minimize certain circuit violations. Circuit metrics may be used to estimate rough circuit solutions when following physical synthesis steps. Common metrics to evaluate circuit solutions are 1) *wire length*, 2) *area density utilization*, 3) *routing congestion*, 4) *timing violations* and 5) *cell displacement*. In the rest of this section, these metrics will be briefly introduced.

3.6.1 Wire Length

In digital circuits, wire segments must be routed in vertical or horizontal directions. The *Manhattan* or norm one distance is a fast and straightforward method to compute the distance between two points. The Manhattan distance is the summation of vertical and horizontal distances between two points $p_0(x_0, y_0)$ and $p_1(x_1, y_1)$. In (3.6), the equation to compute the Manhattan distance is presented.

$$d_M(p_0, p_1) = |x_1 - x_0| + |y_1 - y_0| \tag{3.6}$$

where, x_0 and y_0 are abscissa and ordinate positions of point p_0, respectively. x_1 and y_1 are abscissa and ordinate positions of point p_1, respectively.

The *Euclidean* distance or norm two, is another approach to compute the distance between two points. Euclidean distance is the length of a straight line between two points $p_0(x_0, y_0)$ and $p_1(x_1, y_1)$. In (3.7), the Euclidean equation is shown.

$$d_E(p_0, p_1) = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \tag{3.7}$$

where, x_0 and y_0 are abscissa and ordinate positions of point p_0, respectively. x_1 and y_1 are abscissa and ordinate positions of point p_1, respectively.

In placement, a common metric to measure circuit wire length is the Half Perimeter Wire Length (HPWL). Total circuit HPWL is the summation of the Manhattan distance for all net-bound boxes. In (3.8), the HPWL equation is presented.

$$HPWL(\vec{x}, \vec{y}) = \sum_{e \in nets} [\max_{i \in e} x_i - \min_{i \in e} x_i] + \sum_{e \in nets} [\max_{i \in e} y_i - \min_{i \in e} y_i] \tag{3.8}$$

where, $HPWL(\vec{x}, \vec{y})$ is the total circuit wire length. $e \in nets$ is a circuit net. In Cartesian abscissa, $max\ x_i$ and $min\ x_i$ are the maximum and minimum abscissa positions of cells of net e, respectively. In Cartesian ordinate, $max\ y_i$ and $min\ y_i$ are the maximum and minimum ordinate positions of cells of net e, respectively.

The HPWL metric is easy and fast to compute. However, HPWL is inaccurate when estimating wire length for hyperedge nets. HPWL is convex, continuous and it is not always differentiable (MARKOV; HU; KIM, 2015). In global quadratic placement, HPWL cannot be used to estimate wire length because HPWL is not always differentiable.

Circuit wire length may also be estimated with *monotone chain*, *clique* and *star* net models, or *Steiner tree*. In the monotone chain model, all pins of the net are connected using a Hamiltonian path. In the clique net model, wire length is the summation of the length of all pair connections. In the star model, the source pin is connected to all sink pins with pair connections. The source pin is the central node of the star net model. Wire length is the summation of the length of all pin connections of nets. In the Steiner tree model, wire length is estimated by building the Rectilinear Minimum Spanning Tree (RMST), Rectilinear Steiner Minimum Tree (RSMT), Rectilinear Steiner Arborescence (RSA) or Single-Trunk Steiner Tree (STST) (KAHNG et al., 2011). The Steiner tree method to estimate wire length requires significantly more computational resources compared to the HPWL model.

3.6.2 Area Density Utilization

In global placement, area density utilization is an important metric to measure the quality of the placement solution. Area density utilization indicates cell concentration. Moreover, the area density utilization metric can provide relevant insight into placement algorithms, to find optimized-solutions in regions that have high-cell concentration. Reducing the peak of the area density utilization can also contribute to minimizing cell displacement. Cells must be moved out from high-density regions to minimize area density violations. Several cells must be moved out from high-density regions to alleviate area density violation. Moved cells can be placed far away from the initial position. Therefore, a global placement solution can be significantly modified to fix area density violations. Placement solutions with high-cell concentration can result in infeasible circuit solutions in the CTS or routing steps. Circuit designers limit area density utilization for a maximum ratio utilization between zero and one ($0 < ratio \leq 1.0$). Usually, the area density ratio is limited to 90% of the total area utilization. Therefore, at least 10% of the circuit area is available to be used to insert buffers, tie off cells, tie on cells, and cells in the following physical synthesis steps.

Average Bin Utilization (ABU) (KIM et al., 2012) is a simple and fast metric to estimate area density utilization. ABU is a metric to measure the average of the ratio of area density utilization from overfilled bins. In the ABU, circuit core is split into a grid of regular and uniform bins. By default, the bin size is defined as nine times the height of standard-cells. Bins are sorted by the highest to lowest ratio area utilization, excluding bins that are almost entirely covered by fixed cells or macroblocks.

ABU of overfilled bins is presented in (3.9).

$$ABU_\gamma = \frac{1}{n}\sum_{i=0}^{n}\frac{CA_i}{PA_i} \tag{3.9}$$

where ABU_γ is the average area utilization of $\gamma\%$ of the highest overfilled bins. γ is a set of only overfilled bins defined as 2%, 5%, 10%, and 20% of the valid bins. A bin is valid if its placeable area is at least 20% of the total bin area. n is the number of $\gamma\%$ overfilled bins. CA_i is the cell area of bin i $(0 \leq i \leq n)$. PA_i is the area of bin (i $(0 \leq i \leq n)$ which can be used to place movable cells.

In (3.10), the ratio of average overfilled area for $\gamma\%$ bins and maximum area density utilization threshold is presented.

$$ABU_{\gamma OF} = max\Big(\frac{ABU_\gamma}{\Gamma_{target}} - 1.0, 0.0\Big) \tag{3.10}$$

where Γ_{target} is the maximum bin area utilization threshold $(0 < \Gamma_{target} \leq 1.0)$.

Weighted arithmetic mean area utilization is a representative value to indicate high cell concentration. A high value indicates the circuit placement has regions with high cell concentration. In (3.11), weighted arithmetic mean area utilization (i.e., $ABU\_penalty$) formula is shown.

$$ABU\_penalty = \frac{\sum(K_\gamma \times ABU_{\gamma OF})}{\sum K_\gamma} \tag{3.11}$$

where, K_γ is the ratio weight of overfilled bins as presented in Table 3.1. ABU penalty is a representative weighted arithmetic mean of the exceeded area on overfilled bins. ABU penalty is zero if all bins are free of high-density violation. Otherwise, the ABU penalty is higher than zero to indicate that circuit placement has bins with area density violations.

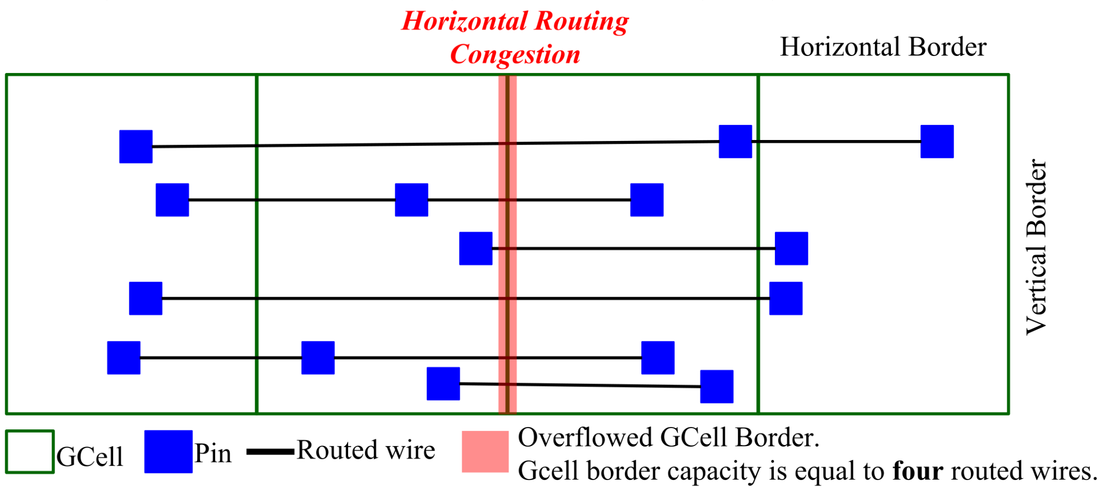
Table 3.1: Weights of average area utilization of γ highest overfilled bins

K_γ	Weight
K_2	10
K_5	4
K_{10}	2
K_{20}	1

3.6.3 Routing Congestion

Routing congestion is an important metric to measure regions which have high routing resource requirements. Placement algorithms can use routing congestion metrics to move cells out from regions with routing violations. The common methods to estimate routing congestion are 1) *congestion map*, 2) *fast global routing*, 3) *pin density*, 4) *Rent's rule*, and 5) *probabilistic methods*. Routing congestion may arise from intra-bin nets, inter-bin nets which one pin is inside of a Global Cell (GCell), and routed nets which just cross GCells (SAXENA; SHELAR; SAPATNEKAR, 2007). Routing congestion metrics have a significant difference between estimated and routed circuit nets (SAXENA; SHELAR; SAPATNEKAR, 2007). Usually, conventional methods to estimate routing congestion use metrics that are fast to compute (SAXENA; SHELAR; SAPATNEKAR, 2007). In Figure 3.8, an example of a GCell grid is presented.

Figure 3.8: An example of GCells. A GCell grid is a grid of bins which covers the circuit core. Each GCell has on each border routing capacity. Routed wires cross GCell border. A GCell grid is a simple and fast way to estimate routing congestion


Horizontal Routing Congestion
Horizontal Border
Vertical Border
GCell
Pin
Routed wire
Overflowed GCell Border.
Gcell border capacity is equal to **four** routed wires.

Source: Author (2019).

A Routing congestion map is determined from a grid of regular and uniform GCells. The circuit core is split into a 2-D $m \times n$ mesh GCells. For each GCell, the maximum allowed number of routed wires to each boundary is computed. Each GCell has routing resource capacity and usage. The GCell capacity indicates the number of wires which can be routed without violations of design rules. GCell capacity is the total routing tracks. GCell usage is the total number of used routing tracks. Routed wires use GCell capacity. GCells have routing congestion if GCell usage is higher than GCell capacity. Routing usage is computed by counting the number of routed wires which cross

GCell boundaries. However, local GCell connections are ignored when computing GCell usage. Routed wires in horizontal that cross vertical GCell boundaries are computed as H-edge. Routed wires in vertical that cross horizontal GCell boundaries are computed as V-edge. Routing overflow is the ratio of GCell usage by GCell capacity . GCells have routing congestion when the ratio is higher than one (LIN; CHU, 2014; WEI et al., 2012).

In GCells, routing usage, capacity and congestion are computed as introduced in (3.12), (3.13), and (3.14) equations, respectively.

$$\begin{aligned} U_g(x,y) = & max(U_{he}(x,y), C_{he}(x,y)) + \\ & max(U_{he}(x-1,y), C_{he}(x-1,y)) + \\ & max(U_{ve}(x,y), C_{ve}(x,y)) + \\ & max(U_{ve}(x,y-1), C_{ve}(x,y-1)) + \end{aligned} \tag{3.12}$$

$$\begin{aligned} C_g(x,y) = & C_{he}(x,y)) + C_{he}(x-1,y) + \\ & C_{ve}(x,y)) + C_{ve}(x,y-1) \end{aligned} \tag{3.13}$$

$$Cong_g(x,y) = \frac{U_g(x,y)}{C_g(x,y)} \tag{3.14}$$

where $U_{he}(x,y)$, $C_{he}(x,y)$ are routing usage and capacity in horizontal edge at (x,y), respectively. $U_{ve}(x,y)$, and $C_{ve}(x,y)$ indicate routing usage and capacity in vertical edge at (x,y), respectively. If any edge (horizontal or vertical) in a GCell located at (x,y) is congested, then this GCell has routing congestion ($Cong_g(x,y) > 1.0$). Congestion maps can be generated using fast global routers or Steiner trees algorithms.

In global routing, nets are routed using global routing algorithms. The objective is to obtain routed net segments to provide a congestion map in relatively low runtime. Global routing techniques are slower than probabilistic methods in generating congestion maps, but they are less pessimistic than probabilistic techniques (SAXENA; SHELAR; SAPATNEKAR, 2007). Global routers such as NCTUgr (HUANG et al., 2015b) can be used to route the circuit and to provide congestion maps.

In the placement algorithms, the pin density metric can be used to estimate routing congestion. Pin Density is the ratio between the number of pins and the area of GCells. In the pin density metric, internal and external pin connections in GCells are captured. On the other hand, the pin density metric fails to capture nets that only cross GCells

(SAXENA; SHELAR; SAPATNEKAR, 2007).

Rent's rule (LANDMAN; RUSSO, 1971) is an empirical routing congestion metrics. The number of IO pins is related to the number of circuit cells, as shown in (3.15).

$$NumPins = CellPins \times NumCells^{r} \tag{3.15}$$

where, $NumPins$ is the number of IO pins. $CellPins$ is the average number of pins of circuit cells. $NumCells$ is the total number of circuit cells. r ($0 \leq r \leq 1$) is the constant Rent's rule that depends on circuit characteristics.

Probabilistic methods are used to measure routing congestion by approximating global routing results. However, probabilistic methods can be inaccurate for estimating the length of nets that are difficult to route. Probabilistic methods also have issues in adequately addressing routing results around complicated routing regions that have routing blockages and macroblocks (SAXENA; SHELAR; SAPATNEKAR, 2007).

3.6.4 Timing Violations

In (KIM; HUJ; VISWANATHAN, 2014; KIM et al., 2015), a metric to measure placement optimization regarding timing violations is presented. The Timing Quality of the Results (TQoR) is a weighted arithmetic mean to compute improvement on timing and area density violations. TQoR metric has been used in the 2014 and 2015 International Conference on Computer Aided Design (ICCAD) contests (KIM; HUJ; VISWANATHAN, 2014; KIM et al., 2015) to evaluate circuit solutions of contest teams. In (3.16), the formula to compute the improvement of timing and area density violations is presented. The improvement in the area density utilization is measured by computing the ABU metric.

$$TQoR = 100 \times \omega_{ABU} \Delta ABU \times \sum_{i \in \{\text{tns, wns}\}} \omega_i \, Q_i \tag{3.16}$$

where ω_{ABU} is ABU (KIM et al., 2012) weight, ΔABU is ABU penalty change from initial placement (i.e., initial minus current ABU penalties). ω_i is the weight for WNS and TNS violations. Q_i is the timing quality score for TNS and WNS. In (3.17), the equation to compute TNS and WNS improvement from the initial solution is presented.

$$Q_{i\in\{\text{tns, wns}\}} = \sum_{j\in\{\text{early, late}\}} \omega_j \left(1 - \frac{i'_j}{i_j}\right) \tag{3.17}$$

where $Q_{i\in\{\text{tns, wns}\}}$ is TQoR for improvement on TNS and WNS violations, respectively. ω_j is the weight for early and late violations of TNS and WNS. i'_j is the current placement TNS and WNS violations while i_j is the initial placement TNS and WNS violations.

In Table 3.2, the 2014 and 2015 ICCAD contests weight factors are presented. The maximum achievable quality score is 1800 points. This score indicates the 2014 and 2015 ICCAD benchmarks are free of timing and area density violations.

Table 3.2: 2014 and 2015 ICCAD contest weights to compute the Timing Quality of the Results

Weight	Value
ω_{abu}	1
ω_{early}	1
ω_{late}	5
ω_{tns}	2
ω_{wns}	1

3.6.5 Driver Strength, Criticality and Centrality

In this section, driver strength, criticality, and centrality metrics are introduced. In circuits with timing violations, driver strength metric aids to determine the direction of cell movement to optimize negative slack. Criticality and centrality are essential to classify critical timing cells by the relevance in the critical timing paths. These metrics are introduced in (FLACH et al., 2016).

3.6.5.1 Driver strength

The driver strength metric indirectly indicates the resistance of the driver cell. In other words, driver resistance can be used to indicate the required time to load the cell output capacitance and the size of the cell. Big cells can load big capacitors faster than small cells.

3.6.5.2 Criticality

Criticality $\{criticality \in \mathbb{R} | 0 \leq criticality \leq 1\}$ is a normalized representative number of the critical timing pins. This number indicates the relevance of the negative slack in the pin compared to the circuit WNS. In Equation 3.18, the pin criticality computation is presented.

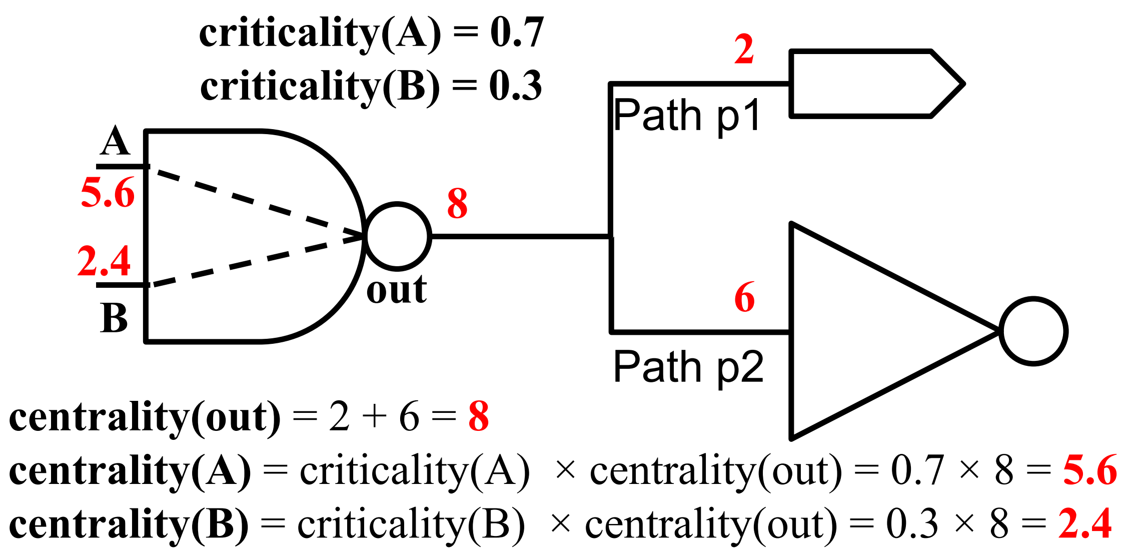
$$criticality(pin_i) = \frac{slack(pin_i)}{\text{WNS}} \tag{3.18}$$

3.6.5.3 Centrality

Normalized centrality $\{centrality \in \mathbb{R} | 0 \leq centrality \leq 1\}$ of a pin is a representative number of the pin's influence in the critical timing paths. Pins with high centrality indicates that worst critical timing paths are passing through these pins.

In Figure 3.9, an example of the centrality computation is presented. the output (out) pin of the nand cell is affected by two critical timing paths (p1 and p2). The centrality of the output pin is the summation of the centrality of the sink pins of the net. The centrality of the input pins (A and B) from the nand cell is the multiplication of the pin criticality and the centrality of the output pin from the nand cell. Pin centralities are normalized in terms of the highest centrality value.

Figure 3.9: Example to compute centrality of critical timing pins



criticality(A) = 0.7
criticality(B) = 0.3
A
5.6
2.4
B
8
out
2
Path p1
6
Path p2
centrality(out) = 2 + 6 = 8
centrality(A) = criticality(A) × centrality(out) = 0.7 × 8 = 5.6
centrality(B) = criticality(B) × centrality(out) = 0.3 × 8 = 2.4

Source: Author (2019).

3.6.6 Cell Displacement

Legalization and detailed placement algorithms may move cells far away from cell positions computed on the global placement solution. These cell movements can significantly impact the quality of the placement solution. Cell displacement may be evaluated using 1) *linear*, 2) *quadratic* or 3) *the relative history* cell displacement models. In linear cell displacement, the distance of current and target positions is computed. In quadratic cell displacement, the quadratic distance between current and target positions is computed. However, these cell displacement models fail to address the history and direction of cell movements regarding initial cell positions. Therefore, it is infeasible to indicate if cell movements are going to increase or to decrease cell displacement by considering only current and target cell positions.

In the model to compute the relative history of the cell displacement (PUGET et al., 2015), the difference of cell displacement is computed considering both current and target positions in terms of the initial position. The difference between both cell displacements gives the relative history of cell displacement. This cell displacement can be a positive or negative value. A negative value indicates the distance between the target and initial positions is lower than the distance between the current and initial positions. Negative values also indicate cells are going to be moved closer to initial positions. These cells have been previously moved away from initial positions Positive values indicate target cell positions are farther away than current positions regarding initial positions. Positive values also indicate cells are going to be moved farther away from initial positions. These cells have been previously moved away from initial positions.

In (PUGET et al., 2015), a cell displacement cost function, which can indicate the history and direction of cell movement, is presented. Equation (3.19) is designed to compute the relative history cell displacement cost as:

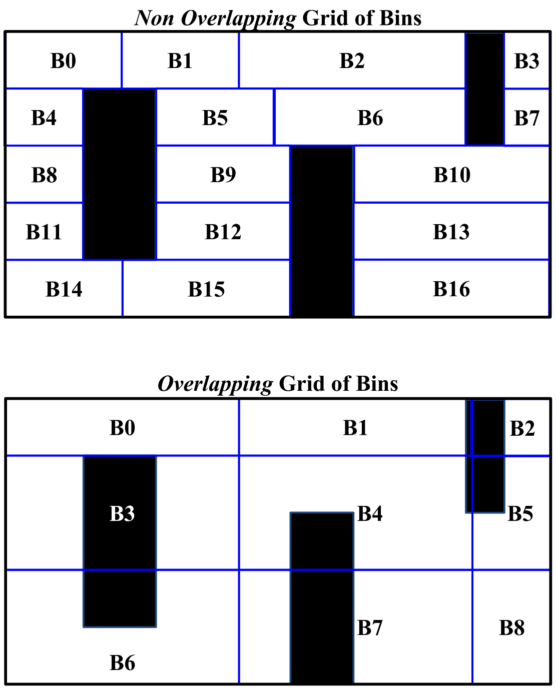
$$cost(.) = |target(.) - init(.)| - |current(.) - init(.)| \tag{3.19}$$

where, $(.) \in \{x, y\}$. $target(.)$ is the position which the cell is going to be moved to. $init(.)$ is the cell position in the global placement solution. $current(.)$ is the actual cell position. The total cell displacement cost is computed as $totalCost = cost(x) + cost(y)$. In (3.19), the signal (negative or positive) of cost value indicates the history and direction of cell movements.

3.7 Grid of Bins for Network Flow-based Cell Spreading Algorithms

In the proposed legalization and detailed placement algorithms, network flow-based cell spreading algorithms rely on a graph which is a grid of bins. A *bin* is a rectangle which covers a small part of the circuit core. Cells are moved out from high-density regions through optimized cost paths. In the graph of the grid of bins, bins are nodes and edges are connections to neighboring bins. Bins which are limited by macroblocks are connected to bins on the opposite side of these macroblocks. Bin connections through macroblocks allow paths outside of regions surrounded by macroblocks. These regions can have area density violations, which only can be solved by moving cells to other regions. In Figure 3.10, two grids of bins are presented. Bins may or may not have overlap with macroblocks or fixed cells.

Figure 3.10: Grids of bins which do not have overlap and have overlap with macroblocks and fixed cells



Source: Author (2019).

The circuit core is split into a grid of bins. This grid can be modeled to overlap or not to overlap macroblocks and fixed cells, as presented in Figure 3.10. In *non overlapping* grid of bins, bin boundaries are limited to row, macroblocks, or fixed cell boundaries. In this model, the bin area is usually small, and bins are free of the fixed area from macroblocks or fixed cells. Bins have the same height and width. Some bins can have width wider than the standard bin width ($standardWidth \leq binWidth <$

$2 \times standardWidth$). In *overlapping* grid of bins, bin boundaries are limited to minimum and maximum row positions in abscissa and ordinate axis. In this model, bins can have overlap to the fixed area from macroblocks and fixed cells. These bins may also have boundaries entirely or partially outside of row boundaries. Bins have the same width and height, except for bins in right and upper circuit core corners. Cells must be placed inside row boundaries. Therefore, some bins may have part or total of the area blocked to place cells because these bins are partially or entirely outside row boundaries.

Area overlap between cells and bins is added to bins as *cell area* component. *Fixed area* is the total area of fixed components which have overlap with bins. *Fixed area* can also be the bin area that is outside of row boundaries. *Bin area* is the total area of the bin. *Place-able area* is the available area to place movable cells inside bins without area density violation. *Target area density* constraint ($0 < targetAreaDensity \leq 1$) is set by the designer to ensure a percentage of the circuit area is always empty. In (3.20), the equation to compute place-able area of bins is presented.

$$placeableArea = (binArea - fixedArea) \times targetAreaDensity \tag{3.20}$$

In non overlapping grid of bins, target area density utilization is defined as 1.0.

Demand and *supply* values are available and overfilled area components in bins, respectively. Therefore, bins with area density violation have the supply value higher than zero. Otherwise, the bin demand area component is higher than zero. Bin *demand* and *supply* are computed in terms of area as shown in equations (3.21) and (3.22), respectively.

$$demand = max\{0, placeableArea - cellArea\} \tag{3.21}$$

$$supply = max\{0, cellArea - placeableArea\} \tag{3.22}$$

3.8 Physical Synthesis

In digital design flow, physical synthesis is composed of several challenging constrained optimization problems. In physical synthesis, objectives and constraints can conflict with each other while optimizing circuit solutions. Usually, optimizing total wire length has a conflict with area density and routability restrictions. Determining a trade-off among conflicting objectives and restrictions can be very challenging. In (3.23), formulation of constrained optimization problems is introduced.

$$\begin{aligned} &Minimize\ f(x) \\ &Subject\ to\ g_i(x) \leq b_i,\ 1 \leq i \leq m. \end{aligned} \tag{3.23}$$

where, $f(x)$ is the objective function to optimize. $g_i(x)$ is the set of circuit constraints. b_i is the set of constraint upper limits.

In physical synthesis, common objectives to optimize are total wire length, path delay, routability, manufacturability, and area density utilization. On the other hand, commonly addressed constraints are technology design rules, electric constraints, and layout geometry restrictions (KAHNG et al., 2011). Constraints in CMOS technology nodes are geometry rules that ensure the correct manufacturing process of circuit layouts. Electric constraints ensure that performance and correct operation of manufactured circuits are achieved. Geometry restrictions are introduced to reduce optimization and synthesis complexity during physical synthesis optimization. These geometric restrictions can include preference direction of routed wires and equal cell heights. Moreover, technology constraints and geometry restrictions also aid minimizing circuit defects during the manufacturing process.

Physical synthesis flow is composed of 1) *floorplan*, 2) *placement*, 3) *CTS* and 4) *routing* stages. In physical synthesis, a wide range of heuristics and formal methods are used to provide optimized circuit solutions. Several formal methods, such as the system of linear equations, Lagrange, Network flow, ILP, LP, and Mixed Integer Programming (MIP) are used to optimize circuit solutions. Heuristics such as area density utilization, maximum arrival timing of paths, delay of cells, paths and nets topologies, paths and nets characteristics, and wire length are used to guide optimization algorithms.

3.9 Summary

In this chapter, the main concepts, models and optimization techniques related to EDA, and physical synthesis are given. Algorithm and graph definitions are presented. Network flow and branch and cut optimization algorithms are introduced. Methods are highlighted to decompose Hyperedge nets. Metrics to evaluate design quality are presented. The graph of grid bins for network flow-based cell spreading algorithms is shown. Physical synthesis optimization objectives and algorithms are presented.

Digital circuits require a considerable effort to design, synthesize, and optimize. The circuit synthesis and optimization depends on heavily formal methods and heuristics. Computing appropriate heuristic data depends on circuit metrics. Circuit metric estimation is very challenging in EDA. Estimation of circuit metrics, heuristics and formal methods are fundamental to optimized circuits.

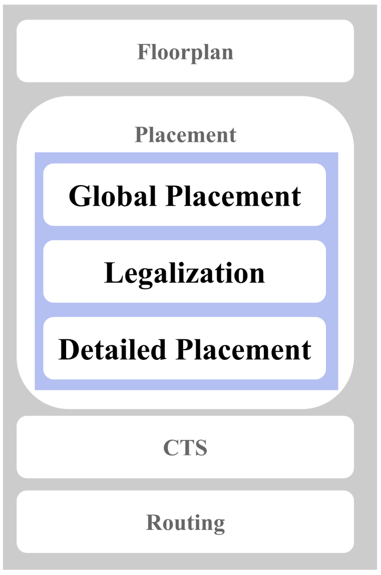
4 DIGITAL CIRCUIT PLACEMENT

4.1 Introduction

Placement is the physical synthesis step that provides optimized cell positions to achieve objectives and to satisfy constraints. Placement can have a significant impact on the quality of the circuit solution. Circuit placement is one of the most critical optimization stages (NAM; CONG, 2007) in physical synthesis. The formulation of the placement problem is NP-hard class. Therefore, obtaining the optimum solution in a reasonable time is infeasible. Besides, advances in CMOS technology processes have introduced a considerable number of complex restrictions to compute optimized cell positions.

Placement stage is divided into 1) *global placement*, 2) *legalization* and 3) *detailed placement*. In Figure 4.1, placement flow is presented. During global placement, approximated cell positions are optimized while cells are allowed to overlap. During legalization, cells are aligned to allocated sites within row boundaries. Cells are also free of cell overlapping. In detailed placement, placement solution is further optimized locally. Usually, detailed placement objectives include minimizing wire length, power consumption, detailed routing violations, and timing violations.

Figure 4.1: Placement optimization flow is composed of global placement, legalization, and detailed placement stages



Source: Author (2019).

Placement algorithms significantly modify cell locations after each algorithm iteration. Therefore, circuit metrics are usually required to be updated for the entire circuit.

Placement strongly relies on heuristics to achieve optimized solutions. Designing placement heuristics is a challenging task. Appropriate heuristic values are strongly related to circuit metrics. Measuring circuit metrics has a trade-off between accuracy and required computational resources. Circuit metrics must be evaluated in a short runtime, be relatively accurate and use the minimum possible computational resources. Also, estimating accurate metrics and heuristics of estimated CTS and routing circuit solutions may be very challenging. The quality of placement solution directly impacts on effort and quality of circuit solutions in CTS and routing steps.

In this thesis, solutions for the proposed legalization and detailed placement algorithms are impacted by the quality of the global placement solution. Therefore, in this thesis, global placement techniques and algorithms are discussed. This chapter is organized as follows: In Section 4.2, global placement optimization objective, and restrictions are presented. In this section, placement algorithms are also highlighted. In Section 4.3, legalization stage and network flow-based legalization algorithms are given. In this section, relevant legalization algorithms are discussed. In Section 4.4, detailed placement optimization objectives and restrictions are given. In this section, detailed placement algorithms are also presented. Finally, in Section 4.5, the chapter summary is given.

4.2 Global Placement

Global placement is a constrained optimization problem. In (4.1), the general formulation of global placement problem is presented.

$$\begin{aligned} &Minimize \sum_{e\,\in\,nets} w_e \times l_e \\ &Subject\ to\ r_i < t_i \\ &\qquad\qquad i \in \{constraints\} \end{aligned} \tag{4.1}$$

where, w_e is weight associated with wire length l_e. $e \in nets$ is a net of circuit. r_i is placement constraint. t_i is constraint upper limit. $i \in constraints$ is placement constraint.

Placement problem formulation is NP-hard class. Therefore, obtaining the optimum solution in a reasonable time is infeasible. Besides, advances in CMOS technology processes have introduced a high number of complex restrictions to comply with during optimization of cell positions. Cell overlapping and alignment to site and row boundaries are relaxed to reduce the complexity of obtaining feasible placement solutions. Moreover, the density of cell area utilization indirectly addresses overlap among cells. Hence, area density indirectly and roughly indicates the total number of cells that are going to be placed in a circuit region. The typical global placement formulation is to optimize total wire length subject to maximum area density utilization. In some formulations routing violations and timing delay are included as part of the placement restrictions. It is assumed that optimized wire length in placement solutions indirectly improves routability, circuit delay, power consumption and violations in design rules.

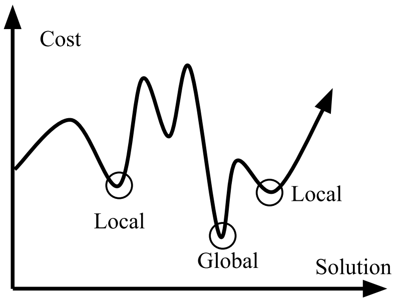
Global placement algorithms may be categorized as 1) *stochastic*, 2) *partitioning-based*, and 3) *analytical*. Stochastic algorithms rely on non-deterministic optimization techniques such as, the Simulated-Annealing (SA) (KHACHATURYAN; SEMENOVSOVSKAYA; VAINSHTEIN, 1981) technique to iteratively improve an initial placement solution. In partitioning-based algorithms, the circuit netlist is recursively split into smaller circuit netlists and assigned to circuit regions. Analytical algorithms rely on numerical optimization methods to provide placement solutions. Analytical placement algorithms may be classified into 1) *non linear* and 2) *quadratic* types. The classification of analytical placement algorithms depends on the numerical method used to compute optimized cell positions. In the quadratic placement, usually, the circuit netlist is modeled as a mass-less spring system. Placement solution is obtained by iteratively minimizing the difference between wire length and density of the upper and lower bound placement solutions.

4.2.1 Stochastic Global Placement

Stochastic placement methods rely on non-deterministic optimization of the constrained objective functions to obtain feasible solutions. SA is a probabilistic method for approximating the global optimum solution in an ample search space for a given objective function. The most used and known stochastic placement algorithms are based on the Simulated-Annealing (SA) (KIRKPATRICK; GELATT; VECCHI, 1983) technique. In Figure 4.2, an example of the SA solution space is presented. In this figure, the SA algorithm is based on navigating through many local solutions trying to reach the global optimum solution. The SA algorithm accepts solutions with high cost to avoid being stuck in a local minimum solution.

The SA algorithm is based on the annealing process from the metallurgy. The SA optimization procedure starts with an initial temperature that has a high value. A cooling function reduces the temperature. An acceptance function determines if worst

Figure 4.2: Solution space of the Simulated Annealing algorithm



Source: (KAHNG et al., 2011)

cell movements are accepted based on temperature value. Cells are randomly chosen to be moved to new positions. Cells are placed in new positions if the objective function is optimized. Cell positions that worsen objective function may be accepted expecting future cell movements to achieve better positions, which further improves the objective function. The SA algorithm accepts some cell movements that worsen the objective expecting to avoid being stuck in a local minimum placement solution. Theoretically, SA may reach an optimal placement solution if it is run for a long time. In the SA approach, the continuous growing in circuit size causes the runtime to become excessive. The excessive runtime prevents SA algorithms to be used in modern circuits (SPINDLER; JOHANNES, 2007). SA algorithms also have poor scalability to optimize circuits. One of the most prominent SA placement algorithms is Timberwolf (SUN; SECHEN, 1993).

In Algorithm 1, a pseudo code of global placement using SA is presented. The SA algorithm receives a mapped netlist and the initial placement solution. The output of the algorithm is a placed netlist with the optimized objective. Usually, the objective is to minimize the total wire length. Temperature, and the number of iterations are initialized in Lines 1, and 2, respectively. The best placement solution is initialized with the initial placement solution (Line 3). In Line 4, the cost of best placement solution is computed. Temperature is commonly initialized with a very high value. The SA algorithm iterates while the maximum number of iterations or the minimum temperatures are not reached (Line 5). At a certain temperature, the SA algorithm iterates over a certain number of cells while a stop condition is not reached (Line 6). This stop condition can be defined as an upper limit of the number of cell movement, an achieved objective value or the number of iterations. A pair of movable cells are randomly selected (Line 7). Cells are temporally switched (Line 8). The cost of switched cells is computed (Line 9). Delta cost of switched cells and current solution is computed (Line 10). A random number ($0 < number < 1$) is found (Line 11). Swapped cells solution is accepted if placement solution is improved

Algorithm 1: Simulated Annealing Placement Algorithm

```
   Data: mapped netlist, initial placement solution
   Result: global placement solution
1  temp = initTemperature();
2  it = 0;
3  bestPlace = initialPlacement;
4  minCost = cost(bestPlace);
5  while $it \leq maxIts\ and\ temp > minTemp$ do
6      while !condition do
7          pair = selectPair(bestPlace);
8          sol = move(bestPlace, pair);
9          cost = cost(sol);
10         $\Delta$ = cost - minCost;
11         val = random(0, 1);
12         if $\Delta < 0\ or\ val < e^{-\Delta/temp}$ then
13             minCost = cost;
14             bestPlace = sol;
15         end
16     end
17     temp = $\alpha\times$ temp;                    /* $0 < \alpha < 1$ */
18 end
19 return bestPlace;
```

(Line 12). Swapped cells which worsen placement solution have a higher chance of being accepted in high temperature. This approach helps SA to avoid getting stuck in local optimum placement solutions. If switched cells solution is accepted, then cost (Line 13) is updated. The best placement solution (Line 14) is also updated. Temperature is reduced by an alpha (α) factor (Line 17). The alpha factor is experimentally determined. The SA algorithm returns the best placement solution (Line 19).

4.2.2 Partitioning-based Placement

In partitioning-based placement algorithms, the circuit netlist is recursively divided into partitions with partitioning algorithms such as Kernighan-Lin (KERNIGHAN; LIN, 1970), Fiduccia-Mattheyses (FIDUCCIA; MATTHEYSES, 1982) or hMetis (KARYPIS; KUMAR, 1998). The objective is to obtain small cell partitions where cells have high connectivity. A cut line function in the circuit graph determines the location to split partitions to evenly distribute cell and circuit areas (SPINDLER; JOHANNES, 2007). The partitioning process stops when the partition reaches the established threshold of the maximum number of cells inside partitions.

In Algorithm 2, a pseudo code of the partitioning-based placement method is presented. The partitioning-based algorithm receives mapped netlist, circuit floorplan, and

Algorithm 2: Min-Cut Placement Algorithm

```
   Data: netlist, floorplan, minimum number of cells per region
   Result: global placement solution
 1 place = NIL;
 2 regions.insert(netlist, floorplan);
 3 while !regions.empty() do
 4     region = regions.pop();
 5     if region.getNumCells() > min_cell then
 6         (r1, r2) = bisection(region);
 7         regions.insert(r1);
 8         regions.insert(r2);
 9     else
10         placeCells(region);
11         place.insert(region);
12     end
13 end
```

the minimum number of cells per partition. The algorithm output is a placed netlist. Placed netlist is initialized with an invalid solution (Line 1). In the partitioning-based algorithm, regions are composed of a set of cells and part of the circuit core. The first region, which is the netlist and floorplan, is inserted in the list of regions (Line 2). The partitioning-based algorithm iterates while there are regions which have the number of cells higher than the threshold (Line 3), i.e., the list of regions is not empty. The next region to partition or place cells is retrieved from the list of regions (Line 4). If this region has more cells than the minimum cell limit (Line 5) then the region is partitioned (Line 6). These partitioned regions are inserted in the list of regions (Lines 7, and 8). Otherwise, cells in this region are placed (Line 10) and the placed region is inserted into the placement solution (Line 11).

4.2.3 Analytical Global Placement

In the global analytical placement, the placement problem is formulated as an analytical cost function to be optimized using numerical methods (SPINDLER; JOHANNES, 2007). Analytical placement algorithms may be categorized into 1) *nonlinear* or 2) *quadratic* depending on the cost function formulation. Nonlinear placement algorithms rely on nonlinear functions (e.g., log-sum-exp) to optimize placement solution. Nonlinear algorithms may require high computational resources to converge the placement solution. These algorithms may be numerically unstable. Usually, nonlinear placement algorithms use a multilevel optimization approach to reduce the complexity to obtain a solution. Some relevant nonlinear placement algorithms are APlace (KAHNG; WANG, 2006), mPL6 (CHAN et al., 2006), NTUPlace3 (CHEN et al., 2008), NTUPlace4 (HSU et al., 2011). Circuit cells and placement area are modeled as electric charges in the ePlace (LU et al., 2014) algorithm. In the ePlace, cells are spread to reach the electrostatic equilibrium.

In the quadratic global placement, the placement optimization problem is modeled as a quadratic cost function. Total wire length optimization is addressed as a quadratic function in quadratic global placement. Quadratic functions are everywhere, differentiated, continuous, and convex. Quadratic function solutions can be obtained by solving the system of linear equations. The force-directed formulation is a category of quadratic placement. In force-directed placement, the netlist is modeled as a spring system from classical mechanics (Hooke's law) (KAHNG et al., 2011). Circuit nets and cells are modeled as springs and mass-less objects, respectively. Hyperedge nets are decomposed into sets of pair connections. Each connection (spring) attracts connected cells. The attraction force is proportional to the quadratic of the distance of cells (KAHNG et al., 2011). The minimum global energy of the spring system is the solution that gives the minimum total wire length in placement. However, this placement solution has a high-cell concentration in small circuit regions. Therefore, repelling or spreading forces must be added to the system of linear equations to minimize high-cell concentration. Quadratic force-directed placement algorithms iteratively optimize cell positions by solving the system of linear equations and adding repelling or spreading the force.

The quadratic force-directed placement approach provides dual placement solutions. *Lower bound placement* is the placement solution obtained by solving the system of linear equations. *Upper bound placement* is the placement solution obtained by spreading out cells from high-density regions. In the quadratic force-directed algorithms, optimized placement solution is obtained when the difference in dual placement solutions is lower than a threshold. Usually, total wire length is the metric which is used to measure the threshold difference of dual placement solutions.

Optimized wire length placement solutions are obtained by solving the system of linear equations. The system of linear equations is formulated as: let n be the number of movable cells. Each movable cell is placed in a Cartesian position (x, y). Placement solution is obtained by solving the system of linear equations to determine optimized positions for movable cells. Cell positions are split into two independent position vectors $\vec{x} = (x_1, x_2, x_3, ..., x_n)$ and $\vec{y} = (y_1, y_2, y_3, ..., y_n)$. Hyperedge nets are decomposed into sets of pair connections. Each pair connects two movable cells i and j. The connection between cells i and j is associated with weight W_{ij}. Weight W_{ij} depends on decomposition net models. Some weight formulations are discussed in (3.3), (3.4), and (3.5) at Section 3.5. In (4.2), pair connection cost in quadratic global placement algorithms is presented.

$$pairCost_{ij} = \frac{1}{2} W_{ij} [(x_i - x_j)^2 + (y_i - y_j)^2] \tag{4.2}$$

where, W_{ij} is the weight of a pair connection. x_i and x_j are pin positions in the Cartesian abscissa of cells i and j, respectively. y_i and y_j are pin positions in the Cartesian ordinate of cells i and j, respectively.

In global quadratic placement, the system of linear equations is modeled with matrix notation. In (4.3), the matrix notation to optimize quadratic placement is presented. The matrix notation is presented only for Cartesian abscissa (X-axis) coordinate.

$$\Phi(x) = \frac{1}{2} x^T Q_x + d_x^T x + constant \tag{4.3}$$

where Q_x is an $n \times n$ symmetric positive definite matrix. d_x is a $n-$dimensional vector. n is the total number of movable cells in the circuit. Let q_{ij} be row i and column j of matrix Q_x for cells a and b, respectively. The negative weight value of pair connection between movable cells a and b are added to matrix Q_x in the positions $q_{i,j}$ and $q_{j,i}$. The positive weight value is added to matrix diagonals $q_{i,i}$ and $q_{j,j}$. If cell a is fixed, the positive weight of pair connection is added to diagonal position $q_{j,j}$ of matrix Q_x. The weight is multiplied by the abscissa coordinate (x position) of cell a, and the multiplication result is added to position j in vector d_x. Nothing is done if both cells are fixed. The matrix Q_y is built in the same fashion.

Matrices Q_x and Q_y are symmetric, sparse, and positive definite. Therefore, algorithms to solve the system of linear equations can exploit these matrix properties to provide a fast solution with minimal usage of computational resources. Movable cells which are mapped to rows and columns results in a symmetric matrix. Matrices Q_x and Q_y are highly sparse because of the local cell connections between a few cells.

In (4.4), the derivative equation from (4.3) is presented. The solution of the system of linear equations is a placement solution with optimized total wire length.

$$Q_x + d_x = 0 \tag{4.4}$$

The system of linear equations to optimize movable cell positions in the abscissa and ordinate axis may be solved in parallel. The solution of the system of linear equations is independent for the abscissa and ordinate axis. Q_x and Q_y matrices are equal if hyperedges are decomposed with the clique, star and hybrid net models. However, matrices Q_x and Q_y are different for decomposed hyperedge nets with B2B net model.

In the force-directed placement algorithms, spreading forces are added to reduce overlap between cells and to move cells out from high-density regions. Spreading forces may be categorized into 1) *constant spreading force*, and 2) *fixed point force*. In constant spreading force, a constant force is added to each cell in the respective index on matrix b after each algorithm iteration. Therefore, the Hessian matrix Q is constant, except if weights of cell connections are updated. In fixed point force, the extra weight (force) is associated with an extra connection between a cell and a fixed point is added to the Hessian matrix Q.

Quadratic placement algorithms have been the subject of research for several years. Several quadratic placement algorithms focus on optimizing total wire length subject to area density utilization. Other quadratic placement algorithms also address routability and timing violations constraints.

4.2.4 Wire Length-driven Global Placement

In wire length-driven quadratic global placement algorithms, the objective is to minimize total wire length subject to area density utilization. Several relevant global placement algorithms are presented as follows. In (QUINN JR., 1975), the first force-directed quadratic placement algorithm is presented. FastPlace (VISWANATHAN; CHU, 2005) is the first quadratic placement algorithm which has significantly outperformed in runtime with practically the same placement quality of the SA and partitioning-based placement algorithms. mFAR (HU; MAREK-SADOWSKA, 2005) addresses fixed points (i.e., pseudo cell positions) as anchors to spread out cells from high-density regions. Kraftwerk2 (SPINDLER; JOHANNES, 2007) relies on the gradient of Poisson potential to determine movable cell positions. In Kraftwerk2, the cell forces are divided into two categories: 1) *hold* and 2) *move* force components. RQL (VISWANATHAN et al., 2007) explores a linearization technique called force-vector modulation to restructure global placement aiming to minimize wire length. RQL also explores density-aware module and wire length-driven local cell spreading techniques to distribute cells evenly in the circuit core. BonnPlace (BRENNER; STRUZYNA; VYGEN, 2008) spreads cells using recursive partitioning of the circuit core. DPlace2.0 (LUO; PAN, 2008) uses cell anchors and wire length linearization strategy to star net model. ITOP (VISWANATHAN et al., 2010) optimizes wire length and timing in global and detailed placement stages. SimPl (KIM; LEE; MARKOV, 2012) iteratively spreads cells by performing fast rough legalization called Look Ahead Legalization (LAL) after solving the system of linear equations. In Polar (LIN et al., 2013), cells are spread with the LAL approach by considering relative cell positions. Polar also splits the LAL regions that have a significant difference between width and height into smaller regions closer to square shapes. In Maple (KIM et al., 2012), the objective is to improve the correlation of placement solution between successive placement optimization steps. Maple combines an unclustering technique with two-tier progressive local refinement. Complx (KIM; MARKOV, 2012) relies on a subgradient primal-dual Lagrange to optimize global placement. In Eh?Placer (DARAV et al., 2016), the target is to optimize the placement solution subject to fence regions, maximum area density, and detailed routing constraints for complex rules in modern CMOS technology nodes.

4.2.5 Timing-driven Global Placement

In modern circuit designs, interconnection delays dominate timing propagation in timing paths (NAM; CONG, 2007). The placement solution can significantly impact the maximum signal delay required to achieve high-performance (NAM; CONG, 2007). Global placement algorithms can place cells in any position of circuit core since objectives are reached and constraints are attended. The focus of wire length-driven placement algorithms is to optimize total wire length without considering timing path characteristics.

In the timing-driven placement algorithms, the objective is to minimize the total wire length subject to area density utilization and circuit timing. Nets of critical timing paths are prioritized during the placement optimization. Critical timing paths and nets are identified with the STA algorithm (MARKOV; HU; KIM, 2015). Timing-driven placement algorithms may also be targeted to optimize WNS, TNS or both timing metrics.

Circuit timing may be optimized during global placement relying on 1) *net-based*, 2) *path-based* or 3) *hybrid* techniques. In net-based techniques, the STA algorithm is used to estimate the timing propagation of electric signals. Net delays are modeled as timing weights (MARKOV; HU; KIM, 2015). Static timing weights remain constant during the entire placement optimization. Dynamic timing weights are updated after each iteration of the placement algorithm. Dynamic weights are gradually updated to reflect path delay changes based on slack or net criticality. Critical timing nets with dynamic weights may oscillate between critical and non-critical timing nets. In path-based methods, the objective is to eliminate timing violations in the entire critical timing paths. However, listing all circuit paths is infeasible. Path-based algorithms are hard to scale when there are several critical timing paths. However, path-based algorithms achieve better results than net-based algorithms (MARKOV; HU; KIM, 2015). In hybrid techniques, circuit delay is optimized with both net-based and path-based techniques.

Some timing-driven global placement algorithms are highlighted as follows. (KONG, 2002; TSAY; KOEHL, 1991; BURSTEIN; YOUSSEF, 1985) are net-based timing-driven placement algorithms. Path-based timing-driven placement algorithms are presented in (PAPA et al., 2008) (WANG; LILLIS; SANYAL, 2005), and (SWARTZ; SECHEN, 1995). In (VISWANATHAN et al., 2010), a quadratic timing-driven placement algorithm is presented. In this technique, critical timing nets are addressed using timing weights. Extra weights are added to the system of linear equations for each critical timing net. These extra weights keep critical timing cells closer when solving the system of linear equations. Critical timing nets can also be addressed in the system of linear equations by adding extra two-pins virtual connections between critical timing cells. In (WU; CHU, 2017), a Lagrangian relaxation timing-driven global placement algorithm is presented.

4.2.6 Routing-driven Global Placement

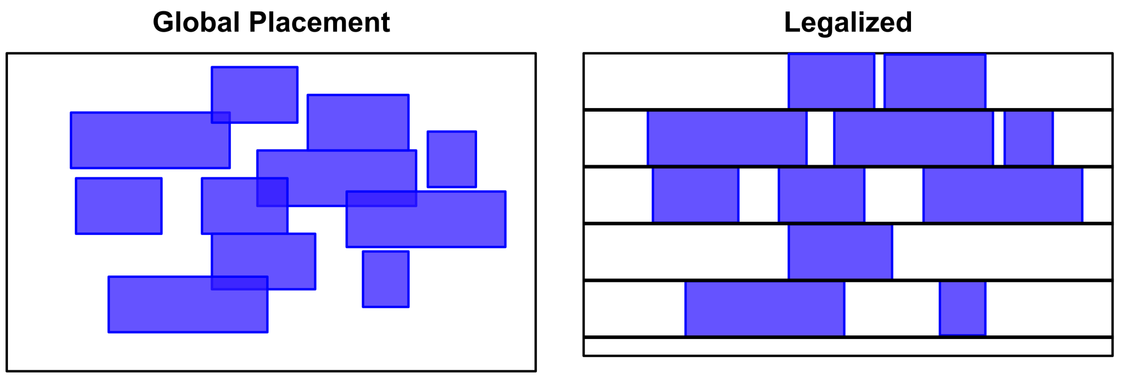
In routing-driven global placement, the objective is to minimize total wire length subject to area density utilization and routing violations. Routing violations are measured with routed nets and congestion map. Routing-driven placement algorithms can also incorporate static or probabilistic techniques to estimate circuit routing congestion. Common techniques to minimize routing congestion are 1) *white space injection* and 2) *cell boating* techniques.

In global placement, several algorithms optimize the total wire length subject to routability. In SimPLR (KIM et al., 2011), cell boating with dynamic cell width adjustment is presented. In Ripple (HE et al., 2011; HE et al., 2013), cells are independently inflated in vertical and horizontal directions. In Ripple 2.0 (HE et al., 2013), cells are moved out from congested regions using paths aware of routability. In NTUPlace4 (HSU et al., 2011), pin density, routing overflow optimization, and porosity of macroblocks are considered to optimized placement solution. In (CONG et al., 2013), large placement regions inside of the circuit, which are free of macroblocks, are identified, and dummy cells are inserted to improve routability. In Polar 2.0 (LIN; CHU, 2014), routing demand is distributed in the rough legalization stage. Routing congestion is independently addressed on the vertical and horizontal directions. In (HUANG et al., 2015b), a clustering algorithm aware of fence regions is presented. A white space insertion for pre-placed pins is shown. In Eh?Placer (DARAV et al., 2016), a global placement algorithm aware of fence regions is shown.

4.3 Legalization

In legalization, cells are placed inside rows and aligned to site boundaries. Legal cell positions are free of cell overlap. During global placement, cell overlap and alignment to row (power rails) and site boundaries are relaxed. Legalization algorithms must modify the placement solution to provide a legal placement solution with minimized cell displacement. Legalization is performed using global placement solutions as the initial placement solution. These solutions need to be evenly distributed and have relatively small cell overlapping (MARKOV; HU; KIM, 2015). In Figure 4.3, examples of global and legalized placement solutions are presented.

Figure 4.3: Examples of global and legalized placement solutions



Source: Author (2019).

Legalization techniques may be classified in terms of 1) *heuristics* or 2) *formal* approaches. Most of the legalization algorithms fall in the heuristic group, such as (HILL, 2002; VISWANATHAN et al., 2007; SPINDLER; SCHLICHTMANN; JOHANNES, 2008a; PUGET et al., 2015), where greedy cell spreading and cell legalization approaches are used. In these algorithms, the objective is to search for the nearest free space from high-density regions to place cells in legal positions. Cells are sorted in a specific order. Cells are placed in legal positions following this ordering. Heuristic legalization algorithms can lead to large cell displacements, which can significantly disturb the global placement solution. High cell displacement can increase wire length, which can degrade circuit performance (VISWANATHAN et al., 2010). Circuit routability can also be adversely impacted by significant cell displacement.

In the formal approaches, legalization algorithms rely on formal methods such as minimum-cost formulations (BRENNER, 2013; CHO et al., 2010; DARAV et al., 2017) or force-directed movements (REN; PAN; KUNG, 2005). Usually, in minimum cost formulation, optimized paths are computed to move cells out from high-density regions. These formal legalization algorithms may have two stages: 1) *cell spreading* and 2) *cell legalization*. During the cell spreading stage, cells are moved out from high-density regions with minimized cost. Also, cell overlap and alignment to site boundaries are relaxed. Cells are only aligned to row boundaries. In the cell legalization stage, cells are placed in legal positions. Cells are placed in aligned positions to row and site boundaries. These placed cells are free of cell overlapping. Formal legalization techniques tend to take a global view of global placement solution.

Legalization algorithms may also be classified as 1) *local* and 2) *global* methods (MARKOV; HU; KIM, 2015). In the local legalization method, cells are legalized in the nearest available positions of the global placement solution. One cell or a small group of cells are legalized in each algorithm iteration. Legalization algorithms have only the stage that directly legalizes cells. High-density regions are eliminated during cell legalization by legalizing cells in the closest positions which have available space to place cells. Some legalization algorithms move already legalized cells to open the required space to legalize a new cell. Cells from regions that have area density violations are moved to neighbor regions that may accommodate these cells free of area density violations. In global legalization methods, usually, legalization algorithms have two stages. Initially, cells are moved out from high-density regions with optimized cell displacement cost formulation. In this stage, cell overlapping and alignment to site boundaries are relaxed. Some legalization algorithms rely on the network flow technique to move cells out of the high-density regions. In the second stage, cells are placed in legal positions. In this stage, circuit placement is free of high-density regions, which aids to improve the quality of legalized placement solution. In global approach methods, cell displacements are generally shorter compared to the local legalization approach.

Heuristic-based and local method legalization algorithms are presented as follows. Tetris (HILL, 2002) is one of the first legalization algorithms. Tetris legalizes cells in the current position or the nearest available right position without moving already legalized cells. Abacus (SPINDLER; SCHLICHTMANN; JOHANNES, 2008a) is similar to Tetris. However, Abacus moves already legalized cells to minimize total cell displacement. Jezz (PUGET et al., 2015) extends Abacus to support mixed-size circuits with blockage regions. Jezz also incrementally legalizes cells. Incremental legalization features can be helpful for detailed placement algorithms.

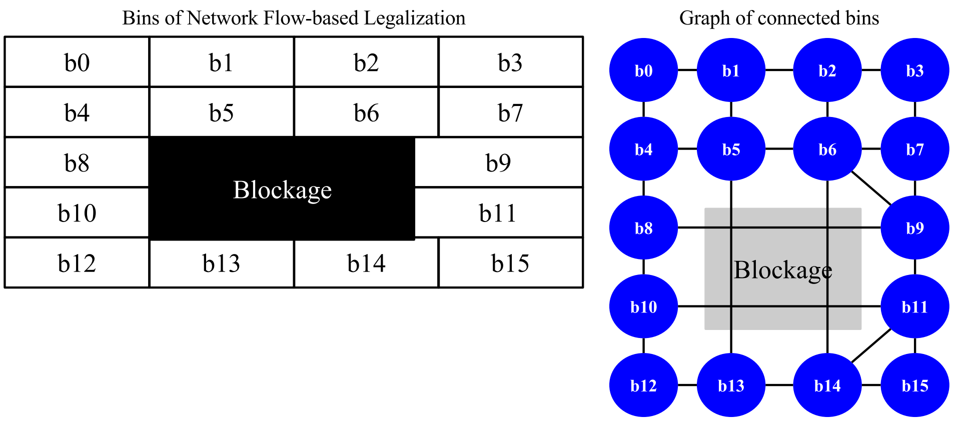
Obstacle-aware Legalization (OAL) (CHOU; HO, 2009) is a legalization algorithm to minimize cell displacement using an exact linear wire length model and obstacle-aware cell insertion. In (HO; LIU, 2010), fast row selection and exact linear wire length model to minimize cell displacement and HPWL techniques are explored. HiBinLegalizer (LEE; WU; CHIANG, 2010) searches appropriate positions to legalize cells in limited area spaces instead of searching in the entire circuit core. HiBinLegalizer also keeps the relative order of cells inside rows. Moreover, HiBinLegalizer uses a weighted sum of Manhattan distance movements to minimize cell displacement. (HU et al., 2013) shows a legalization algorithm that sort cells based on 1) width, 2) pin, and 3) center in the distribution, invalid relocation, and overlap removal steps, respectively. The objective is to minimize total wire length. In (NETTO et al., 2016b), standard-cells are legalized relying on fast queries on multidimensional trees. In (NETTO et al., 2016a), a legalization algorithm that explores parallelism to reduce legalization runtime is shown. In (CHOW; PUI; YOUNG, 2016), a multi-deck legalization technique is presented. In (WANG et al., 2017), a legalization algorithm that explores the difference of cell heights in multi-deck legalization to minimize dead space is shown.

4.3.1 Network Flow-based Legalization Algorithms

Network flow-based legalization algorithms rely on a network flow technique to provide legalized placement solution. Network flow-based legalization algorithms are split into 1) *cell spreading*, and 2) *cell legalization* stages. In the cell spreading stage, cells are moved out from high-density regions with minimized cell displacement cost. Therefore, cell distribution in the placement area is smoother. In the cell legalization stage, cells are placed in legal positions and aligned to site and row boundaries without cell overlapping.

In the cell spreading stage, a grid of bins is built. Bin connections may be modeled as a graph, where nodes represent bins and edges represent connections to neighbor bins. Rows are divided into segments which are limited by row boundaries or by row blockages. Row segments are split into bins which have the same width. Bins are merged with the left bin if bin width is lower then the default bin width. Bin width can be computed as introduced in (BRENNER, 2013). A Bin is connected to all neighbor bins and bins on the opposite side of macroblocks or placement blockages. In Figure 4.4, the grid of bins and the representative graph are presented. All movable cells are partially or entirely inserted to respective bins that have cell overlap.

Figure 4.4: Bins are connected to their immediate neighbors or the bins on the opposite side of macroblocks or blockages. Connected bins are modeled as a graph



Source: Author (2019).

Demand and *supply* values are computed in terms of area for each bin. Demand is the amount of available area to place cells in bins before bin capacity is reached. Supply is the amount of area that exceeds bin capacity. Bin capacity is the total area of the bin. In (3.21) and (3.22) (Section 3.7), demand and supply equations are introduced, respectively.

In the grid of bins, optimized cost paths are computed to move cells out from high-density regions. In Algorithm 3, a pseudo code of network flow-based legalization algorithm is presented. The algorithm inputs are netlist, global placement solution and circuit floorplan. Legalized placement solution is the algorithm output. The first algorithm procedure is to build the grid of bins and to compute demand and supply areas

Algorithm 3: Network Flow-based Legalization Algorithm

```
Data: circuit netlist, global placement solution, and circuit floorplan
Result: legalized placement solution
1  buildGridOfBins();
2  OFBins = computeOverfilledBins();
3  while $!OFBins.empty()$ do
4      for $bin \in OFBins$ do
5          path = pathAugmentation(bin);
6          if valid(path) then
7              moveCells(path);
8          end
9      end
10     OFBins = computeOverfilledBins();
11 end
12 legalize();
```

(Line 1). The list of overfilled bins is obtained (Line 2). The path augmentation algorithm is iterated while there are overfilled bins (Line 3). For each overfilled bin, the path augmentation algorithm is executed (Lines 4 and 5). The path augmentation procedure may be implemented with Dijkstra (DIJKSTRA, 1959) or Breadth-First Search (BFS) (CORMEN et al., 2009) techniques. In valid paths (Line 6), cells are moved (Line 7) between bins. In generalized network flow, optimized paths can be infeasible. Outflow may be higher than bin capacity for all leaf nodes of candidate paths. After all overfilled bins are visited, remaining overfilled bins are obtained (Line 10). Only one iteration of the path augmentation algorithm may be enough to fix the area density violation in bins. The last procedure is to place cells in legal positions and align cells inside of the site and row boundaries.

In the network flow-based legalization algorithms, the cost function is essential to select cells to move out from high-density regions. Appropriate cost function has an influence on cell displacement. Cell displacement can be modeled with linear or quadratic functions. However, both cost functions are unable to address the history and direction of cell movements based on the signal of cost values. The history and direction of cell movements are essential to determine if moved cells are going to increase or decrease cell displacement.

Network flow-based legalization algorithms are presented as follows. Domino (DOLL; JOHANNES; SIGL, 1991; DOLL; JOHANNES; ANTREICH, 1994) is one of the first network flow-based legalization algorithms. Domino legalizes the circuit by solving a network flow formulation of same-size sliced-cells. In the final step of the Domino algorithm, cell pieces are reassembled. In (CHO et al., 2010), history-based in the network flow formulation to legalize cells is integrated. In (BRENNER, 2012) and BonnPlace Legalization (BRENNER, 2013), a network flow technique is developed to reduce cell displacement and to move cells out from regions that have area density violations. BonnPlace Legalizer computes the path of bins to move precise cell flow area from source to sink bins. The BonnPlace algorithm computes path flow bins from a source bin using the shortest path Dijkstra (DIJKSTRA, 1959) algorithm. Eh?Legalizer (DARAV et al., 2017) algorithm computes path augmentation using the BFS (CORMEN et al., 2009) technique.

4.4 Detailed Placement

In detailed placement, the placed netlist is locally optimized. Usually, the focus of detailed placement algorithms is to optimize a single objective at a time. The typical optimization objectives are 1) wire length, 2) timing violations, 3) power consumption, 4) routability, and 5) manufacturability. Recently, detailed placement algorithms also minimize violations of design rules from detailed routing.

In the detailed placement algorithms, a single cell or a small group of cells are moved to optimized positions in each iteration. Usually, the input of the detailed placement algorithm is a legalized placement solution. The placed netlist may be legalized after detailed placement optimization. In this approach, detailed placement algorithms optimize circuit objectives in an illegal placement solution. A legalization algorithm is required to legalize placement circuit afterward. Therefore, the legalization procedure can have adverse side effects on the optimized placement solution. On the other hand, legalizing each cell movement in detailed placement algorithms requires a dedicated legalization procedure in the optimization algorithm or an incremental legalization algorithm.

Detailed placement algorithms rely on heuristics and formal methods to compute optimized cell positions. Formal methods include mathematical techniques such as Lagrange multipliers, the system of linear equations, analytic equations, linear programming, ILP, LP, and MIP. Characteristics of paths, nets, and cells are also considered when optimizing placement solution in detailed placement. Path characteristics can be buffer chain, fixed cells, and macroblocks. Net characteristics may be the number of sink cells, and critical timing sink cells. Cell characteristics can be driver strength, criticality, and centrality.

Several detailed placement algorithms have been proposed to optimize the placement solution in modern circuit designs. The objective of these algorithms is to optimize the placement solution subject to complex design rules. In (DU; WONG, 2014), the objective is to optimize detailed placement for complex $16\,\mathrm{nm}$ FinFET design rules. DF-Placer (HAN; KAHNG; LEE, 2015) algorithm optimizes the detailed placement solution subject to complex sub-$14\,\mathrm{nm}$ constraints. In (DOBRE; KAHNG; LI, 2015), the objective is to improve the physical design layout by addressing mixed cell height in advanced technology nodes. The MrDP (LIN et al., 2016) algorithm moves heterogeneous-sized cells to optimize positions where is improved bot wire length and area density utilization. In (WU; CHU, 2016), wire length is optimized by transforming mixed-height cells into cells with the same height.

4.4.1 Wire Length-driven Detailed Placement

The focus of wire length-driven detailed placement algorithms is to minimize total wire length. Cells are moved to optimized positions where the total wire length of connected nets can be minimized. Usually, wire length is measured with the HPWL metric. Detailed placement algorithms that minimize wire length are presented as follows.

In (PAN; VISWANATHAN; CHU, 2005), several local optimization techniques to minimize wire length are presented. The techniques are 1) to move cells to the optimum region with a global swap algorithm, 2) vertical cell swap, 3) local cell reorder, and 4) cell cluster in single segments. In (CONG; XIE, 2006; CONG; XIE, 2008), a three-step optimization flow called XDP is presented. The objective is to minimize wire length in mixed-size detailed placement. In the (CONG; XIE, 2006; CONG; XIE, 2008) approach, a combination of constraint graph and linear programming to legalize macros, an enhanced greedy method to legalize standard cells and a sliding-window-based cell swapping to further reduce wire length are presented. ECO-System (ROY; MARKOV, 2007a; ROY; MARKOV, 2007b) integrates several incremental techniques to minimize wire length. ECO-System feature is white space redistribution in the circuit with fixed cells and macroblocks. ECO-System relies on cell swapping, greedy legalization, linear programming, network flow, and sliding window techniques to minimize total wire length. In (CAULEY et al., 2011), MIP and branch-and-cut strategies to minimize wire length and cell displacement are presented. In (LI; KOH, 2012), two MIP models to minimize wire length and via count are shown. In (WARD et al., 2013), the optimized alignment of embedded data paths at global and detailed placement stages are given. BraveDP (POPOVYCH et al., 2014) presents an aggressive cell swapping technique to minimize wire length. In (CHOW et al., 2014), global and local cell movements to minimize wire length subject to preserving global placement solution, cell displacement constraint, and maximum cell density area are presented. In (ZHOU; HU; ZHOU, 2014), a cell swapping technique to minimizing wire length subject to maximum cell density is shown.

4.4.2 Timing-driven Detailed Placement

The focus of timing-driven detailed placement algorithms is to compute optimized timing positions to move cells to minimize timing violations. Circuit timing and critical timing paths are evaluated using STA technique. Detailed placement algorithms prioritize critical timing cells in critical timing paths and nets to compute optimized positions.

Several timing-driven detailed placement algorithms are presented as follows. RUMBLE (PAPA et al., 2008) is a linear-programming-based timing-driven placement. RUMBLE improves timing propagation by enhancing the location of buffers and latches for optimum timing propagation. Quadratic placement techniques (VISWANATHAN et al., 2010; MONTEIRO et al., 2015; FOGAçA et al., 2016) can also optimize timing violations. Extra weights are added to the system of linear equations for each critical net. Extra weights keep critical timing cells closer when solving the system of linear equations. Critical nets can also be addressed in the system of linear equations by adding extra virtual connections of two-pins between critical cells in critical timing paths and nets.

Several timing-driven detailed placement algorithms have been proposed using the infrastructure that has been provided in two contest editions in incremental timing-driven placement (KIM; HUJ; VISWANATHAN, 2014; KIM et al., 2015). In these techniques, wire load capacitance is balanced by placing cells nearest to the local optimum positions to reduce path delay locally. The focus of timing-driven placement algorithms is to balance wire load capacitance in nets that contain the driver and sink cells of critical timing nets.

Some timing-driven placement algorithms explore useful clock skew in local clock networks to optimize timing propagation in critical timing paths. In (GUTH et al., 2015), timing violations are optimized with a relaxed Lagrangian formulation. In this formulation, Lagrangian multipliers indicate weights of critical timing nets. In (BOCK et al., 2015), timing violations are optimized with timing path straightening and cell clustering movement techniques. In (BOCK et al., 2015), Euclidean distance of outer pins in critical nets is shortened to reduce wire length and timing propagation in critical timing nets. In (LIVRAMENTO et al., 2015), Steiner tree branches which are free of timing violations are shortened to minimize wire load capacitance of critical timing nets. In (FLACH et al., 2015), a timing-driven detailed placement flow composed of a quadratic timing-driven placement algorithm and useful clock skew optimization is presented. In (LIVRAMENTO et al., 2016), optimized timing positions of registers are computed to explore useful clock skew in critical timing paths. In (FLACH et al., 2016), several techniques to optimize circuit delay incrementally are presented. These techniques rely on net and path topologies and driver cell strength characteristics to compute optimized-timing positions for critical timing cells. In (HUANG et al., 2016), the objective is to minimize early timing violations with slack compression. In (JUNG et al., 2016), a timing-driven detailed placement algorithm with gate sizing and layer assignment flow is shown to optimize circuit timing violations.

Recently, several discrete cell selection algorithms (FLACH, 2015; REIMANN, 2016; FLACH et al., 2014; LIVRAMENTO et al., 2014) have been presented. These algorithms are based on Lagrangian relaxation to optimize circuit timing violations and power consumption. In discrete cell selection, library-cells associated with circuit cells are changed to a different library-cell version that improves the objective. In placement, discrete cell selection requires, in some cases, to open horizontal space to legalize wider cell instances than the previous ones. In buffer insertion, long wires are split into segments by inserting buffers (HU; LI; ALPERT, 2009). The objective is to reduce resistance and capacitance in long wires.

4.4.3 Routing-driven Detailed Placement

Routability is an essential requirement to be addressed in detailed placement. In advanced CMOS technology nodes, routability has become a challenging objective to be achieved. Especially in detailed routing, reduced technology geometry dimensions impose a high number of complex rules to route nets. Common detailed routing violations can be summarized as pin access, short circuit, proximity metal geometry violations, and Non Default Rule (NDR) violations. Neighboring cells can also be affected if pins of cells have routing violations. Nowadays, routing-driven detailed placement algorithms must optimize placement solutions subject to complex design rules to route circuits.

Routing-driven detailed placement algorithms are presented as follows. CROP

(ZHANG; CHU, 2009; ZHANG; CHU, 2013) addresses routing-driven detailed placement via refined and directional module shifting. CROP also assigns weights to wires in congested regions. In (CHOW et al., 2014), circuit routability is indirectly optimized with pin and cell density maps. In (KENNINGS; DARAV; BEHJAT, 2014), detailed routability issues (e.g., pin shorts, pin access, and the minimum spacing requirement) are optimized. In (LI; KOH, 2014), wire length and via count are optimized for detailed routing using large sliding windows with modified MIP. In (HUANG et al., 2015a), a detailed-routing-aware white space allocation technique to minimize detailed routing violations is presented. In (HUANG et al., 2015a), the routability is also optimized with a multi-stage congestion-aware cell spreading technique. The DrDp (TABRIZI et al., 2015) algorithm is a detailed routing-aware detailed placement algorithm with the focus of minimizing detailed routing violations.

4.5 Summary

Placement is a challenging optimization stage. Placement problem formulation and the considerable number of design constraints make the problem very hard to obtain optimized placement solutions. The quality of placement solution has a significant impact on the effort and the quality of solutions during the CTS and routing stages. Placement optimization is split into global placement, legalization and detailed placement. In global placement, optimized cell positions are computed with relaxed cell overlapping and cell alignment to site and row boundaries. In legalization, cells are placed in positions which cells are free of overlap and aligned to site and row boundaries. In detailed placement, placement is locally optimized by moving a single or a small set of cells to optimized positions.

In placement, several formal techniques are used to compute optimized placement solutions. Placement algorithms also depend heavily on circuit heuristics. Placement heuristics are obtained by estimating circuit metrics. Placement algorithms and techniques provide optimized placement solutions subject to numerous restrictions from circuit, design, and CMOS technology node.

5 ROUTING-AWARE INCREMENTAL TIMING-DRIVEN PLACEMENT

5.1 Introduction

Usually, in detailed placement, timing and routability optimization are independently addressed. Timing-driven placement algorithms can move cells to optimized positions. These optimized-positions can be in regions with Routing Overflow (RO). The improvement in timing violations can be reversed to minimize routing violations. On the other hand, routing-driven detailed placement algorithms can optimize routability by severely worsening delay of paths. A detailed placement flow which alternately addresses timing and routability can have cases where optimizing one objective revert improved results of another objective and vice versa.

In this chapter, the routing-aware incremental timing-driven placement contribution is presented. The objective in the contribution is to ensure that the timing requirements of all critical timing nets are met while also ensuring that the routability is not affected. The proposed Routing-aware Incremental Timing-driven Placement (RAITDP) algorithm moves critical timing cells to local optimized positions, which are in regions free of RO. The proposed algorithm relies on net and path topologies to compute optimized timing positions. In each optimized position, timing propagation is locally improved. The summation of local timing improvements minimizes delay violations in critical timing paths. The main contributions in this chapter can be summarized as follows:

- Developing a timing-driven placement optimization flow where the cell movements are subject to routability.
- Determining the optimized maximum cell displacement constraint while considering RO and timing violations.
- Conducting a comprehensive set of experiments by modifying the maximum cell displacement constraint and available routing resources to analyze different aspects of RO and timing violations.

The work presented in this chapter has been published in conference *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* with title **Routing-Aware Incremental Timing-Driven Placement**.

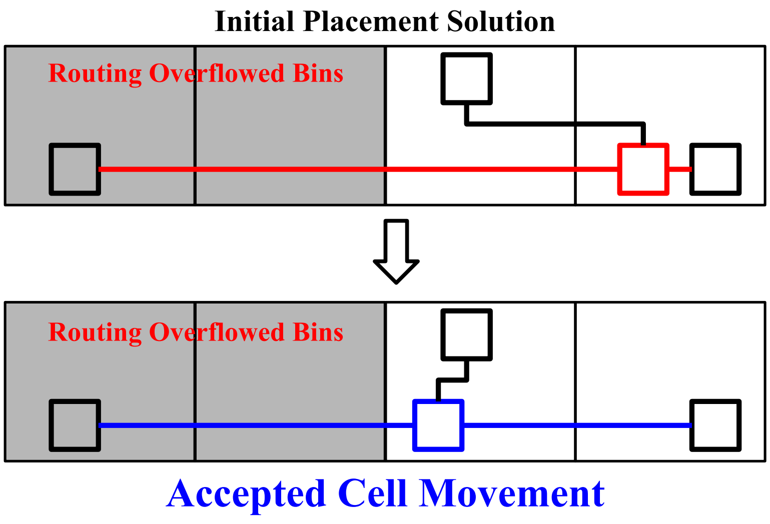
This chapter is organized as follows: The proposed RAITDP algorithm and Incremental Timing-Driven Placement (ITDP) algorithm are summarized in Section 5.2. In Section 5.3, experimental setup and numerical results are presented. Final remarks are given in Section 5.4.

5.2 The Proposed Routing-aware Incremental Timing-driven Placement Optimization Algorithm

In this section, the proposed RAITDP algorithm is presented. The objective of the proposed algorithm is to optimize timing violations in detailed placement, subject to routability, and maximum cell displacement constraints. In the proposed algorithm, the first procedure is to compute optimized positions to minimize the local delay of critical timing cells. The next procedure is to evaluate the RO in optimized-timing positions. These procedures are performed for each optimized-timing position. A cell is moved to the optimized position if this position is in a region with lower RO than RO in the current cell position. Routability restrictions are relaxed if the current and optimized positions are inside of the same GCell.

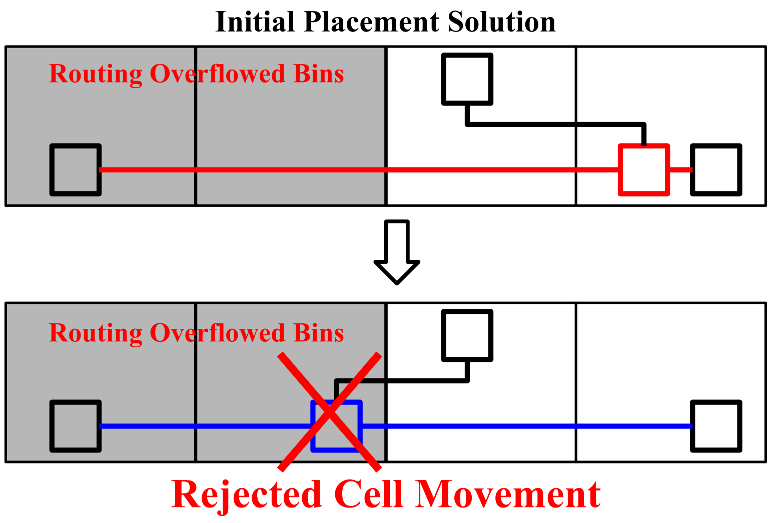
In Figures 5.1 and 5.2, the insight of the proposed RAITDP algorithm is presented. The cell movement to the target position is accepted if the timing violation is minimized and the position is in a region free of RO (Figure 5.1). Cell movements are also accepted if both current and target positions are inside of the same GCell. On the other hand, all cell movements are rejected if the target position is in regions which the RO is higher than the RO in the current positions, as shown in Figure 5.2.

Figure 5.1: In the proposed RAITDP algorithm, timing critical cells are moved to positions which timing violation is improved and RO is lower than the RO in current positions



Source: Author (2019).

Figure 5.2: In the proposed RAITDP algorithm, cell movements are rejected if target positions have RO. Routability restriction is relaxed if both optimized and current cell positions are inside of the same GCell

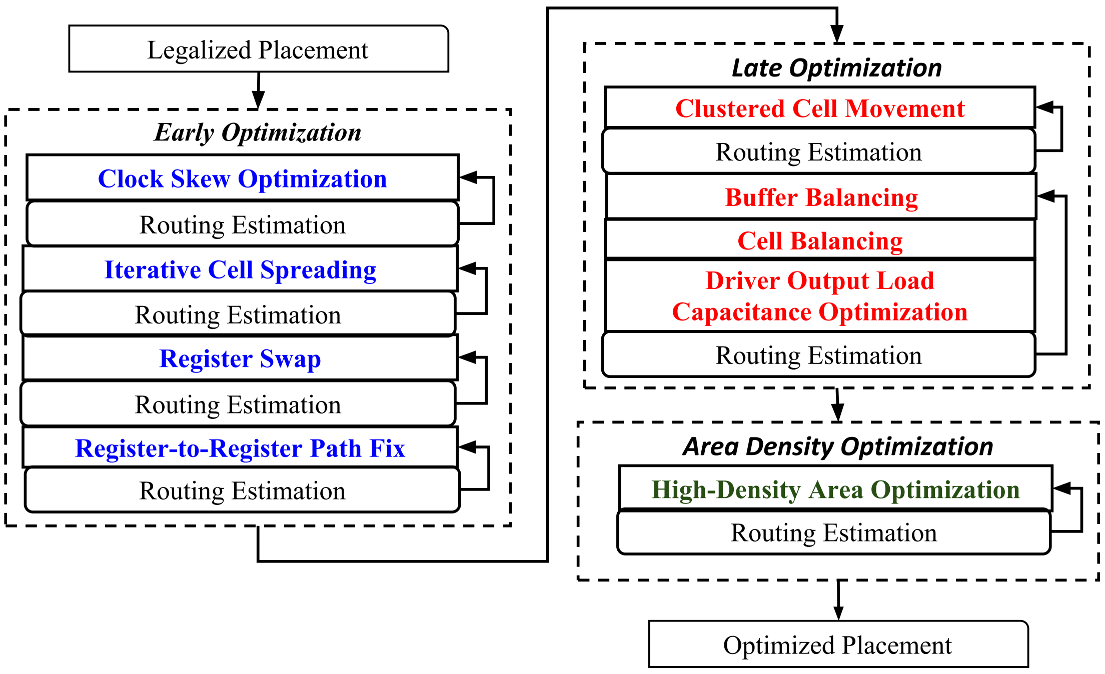


Source: Author (2019).

The proposed RAITDP algorithm relies on a set of incremental timing-driven placement algorithms to optimize early and late negative slacks and high-density regions subject to routability. The RAITDP optimization flow is composed of three stages: 1) *early optimization* (Section 5.2.2), 2) *late optimization* (Section 5.2.3), and 3) *area density optimization* (Section 5.2.4), as shown in Figure 5.3 with dashed rectangles. In the optimization flow, the first two stages are to optimize early and late timing violations, respectively. The last stage is to optimize area density violation. In the proposed RAITDP, optimization algorithms are interleaved with global routing. In late optimization flow, three optimization algorithms are aggregated in a local optimization flow. In this local optimization flow, RO is evaluated only after each iteration of this local optimization flow. The result of the proposed RAITDP is a placement solution with improvement in timing and routing violations.

Placement solution is routed with a global router after each algorithm iteration. Each algorithm iteration moves a set of cells to optimized-timing positions subject to routability and maximum cell displacement constraints. If the placement solution after the algorithm iteration has worsened the Quality of Results (QoR) or RO, then the current placement solution is reverted to the previous placement solution. The algorithm iteration is stopped if the QoR is worsened or the QoR improvement is lower than a threshold value. Iteration of the optimization algorithm is stopped if RO is increased over a threshold. This rollback approach avoids making the placement solution worse and stacks up in an

Figure 5.3: The Proposed Routing-aware Incremental Timing-Driven Placement Flow. Critical timing cells are moved to the local timing-optimized positions subject to routability and maximum cell displacement constraints



Adapted from: (MONTEIRO et al., 2016).

algorithm for a long time without significant placement improvement.

In the proposed RAITDP flow, the circuit is kept legal during critical timing cell optimization. The nearest space which can accommodate a cell is searched from the target position. Legal cell position must be inside of the maximum cell displacement boundaries. In the legal position, the routability constraint must be attended. Legalization procedure is performed with the Jezz (PUGET et al., 2015) algorithm.

In the rest of this Section, these three stages of placement improvement are further discussed. The outline of the proposed RAITDP algorithm is given in Section 5.2.1. In Section 5.2.2, early incremental timing-driven placement algorithms are presented. Late incremental timing-driven placement algorithms are shown in Section 5.2.3. The algorithm to optimize area density utilization is highlighted in Section 5.2.4. Finally, the algorithm to search the nearest white spaces from optimized-timing positions is presented in Section 5.2.5.

5.2.1 Algorithm Outline

In the proposed RAITDP algorithm, critical timing cells are moved to optimized timing positions subject to routability and maximum cell displacement constraints. Cell movements are accepted only if, in the target positions, RO is lower than RO in the current cell positions. The routability restriction is relaxed if both current and target positions are inside of the same GCell. In Algorithm 4, the outline of the proposed RAITDP algorithm is presented. In (FLACH et al., 2016; FLACH, 2015), the flow ITDP to optimize only timing violations is presented. In this work, the objective is to optimize timing violations subject to routability. Therefore, optimized-timing cell positions which are computed by this optimization flow are also subject to RO restriction. RO is also evaluated in the placement solution. The proposed RAITDP algorithm receives a legal placement solution with timing and routing violations. The output is a legal placement solution with minimized timing and routing violations. Optimization algorithms are iterated while the timing violation improves by at least 0.5%, and the RO is lower than 500 thousand ROs. Otherwise, the algorithm iteration is stopped. The placement solution is reverted to the previous placement solution if timing or routability violations are worsened.

In Line 1, STA timer and Jezz are initialized. During initialization, the initial QoR and RO are computed. In this stage, the initial placement solution is stored as the previous placement solution. In Lines 3, 6, 9, 12, 15, 18, 19, 20 and 23, optimization Algorithms 5, 6, 7, 8, 9, 12, 13, 14 and 15 of the proposed RAITDP flow are executed. In Lines 4, 7, 10, 13 and 16 QoR and RO are evaluated for each iteration of optimization algorithms 5, 6, 7, 8 and 9, respectively. In Line 21, QoR and RO are evaluated for each iteration of the flow composed of Algorithms 12, 13 and 14. Finally, in Line 24, QoR and RO are evaluated for each iteration of Algorithm 15. If QoR or RO are degraded, then placement solution is reverted to the previous placement solution. Otherwise, the previous placement solution is updated to the current placement solution.

5.2.2 Early Optimization

The objective of early optimization is to minimize early negative slack subject to maximum cell displacement and routability constraints. Data signals arrive in end points before the early RAT. Therefore, in these end points, early slack is negative. In the early optimization flow, data signals are delayed by increasing wire capacitance and resistance

Algorithm 4: RAITDP Outline

```
Data: Legal placement solution with timing and routability violations
Result: Legal placement solution with improvement in timing, area density
        utilization and routability
1  initialize();
2  do
3  |  clockSkewOpto();              /* Call to Algorithm 5 */
4  while improvedPlacementSolution();
5  do
6  |  iterativeCellSpreading();     /* Call to Algorithm 6 */
7  while improvedPlacementSolution();
8  do
9  |  registerSwap();               /* Call to Algorithm 7 */
10 while improvedPlacementSolution();
11 do
12 |  reg-to-regPathFix();          /* Call to Algorithm 8 */
13 while improvedPlacementSolution();
14 do
15 |  clusteredCellMovement();      /* Call to Algorithm 9 */
16 while improvedPlacementSolution();
17 do
18 |  bufferBalancing();           /* Call to Algorithm 12 */
19 |  cellBalancing();             /* Call to Algorithm 13 */
20 |  driverLoadCapOpto();         /* Call to Algorithm 14 */
21 while improvedPlacementSolution();
22 do
23 |  areaDensityOpto();           /* Call to Algorithm 15 */
24 while improvedPlacementSolution();
```

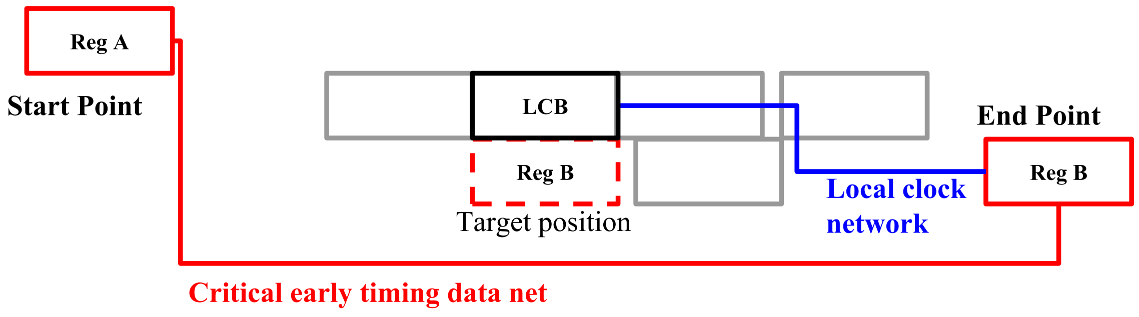
in critical timing nets. Wire capacitance and resistance are increased by moving away from critical timing cells with early timing violation. In this optimization flow, useful clock skew is explored to minimize early timing violation in registers of critical timing paths. Early optimization flow is composed of a set of four algorithms: 1) *Clock Skew Optimization* (Section 5.2.2.1), 2) *Iterative Cell Spreading* (Section 5.2.2.2), 3) *Register Swap* (Section 5.2.2.3), and 4) *Register-to-Register Path Fix* (Section 5.2.2.4).

5.2.2.1 Clock Skew Optimization

In the *Clock Skew Optimization* algorithm, the objective is to move end point registers with early negative slacks closer to the Local Clock Buffer (LCB) of the local clock network to reduce clock latency. Early timing violations can be minimized if the clock latency to capture data signal in the end point register is reduced. However, there might be timing adverse side effects on other data paths when clock latency is changed in registers. Usually, a register is the start and the end points for different data paths. Improving early timing violation in a data path can worse early timing violation in other data paths. Data path timing propagation is locally updated after each register movement to detect timing violation issues in other data paths. Only cell movements which do not have adverse side effects on third party paths are accepted.

In Figure 5.4, an example of the clock skew optimization to minimize early timing violation is presented. *Register B (Reg B)* is an end point cell which has early timing violation. This cell is moved (dashed rectangle) closer to its LCB to reduce clock latency. As the clock latency is reduced, the clock capture transition occurs early in *register B*. Therefore, the data signal in the input of *register B* can be correctly captured.

Figure 5.4: Exploring clock skew to minimize early timing violation. End point register which is the end point of an early timing violation path is moved closer to its LCB. In this approach, clock latency is reduced in the end point register



Source: Author (2019).

In Algorithm 5, the pseudo code of clock skew optimization algorithm is presented. In (FLACH et al., 2016), this algorithm is presented only to optimize timing violation. In this work, the objective is to optimize timing violations subject to routability. Therefore, optimized-timing cell positions which are computed by this algorithm are also subject to RO restriction. This algorithm receives a legal placement solution with timing violation. The algorithm returns a legal placement solution with optimized early negative slack. In this algorithm, end point registers of early timing violation paths are moved closer to their LCB.

In Line 1, end point registers of paths with early timing violation are queried. An end point is a 1) register or a 2) primary output. All end points are visited (Line 2). An end point is valid if 1) it is a register, 2) it is movable, and 3) the clock pin is connected to a local clock network (Line 3). The local clock network is obtained from the valid end point (Line 4). In ICCAD 2015 contest benchmarks, the clock signal of end point registers is provide by LCBs. A LCB is limited to connect up to 50 registers. LCB is the driver

Algorithm 5: Clock Skew Optimization

```
Data: Legal placement solution with early timing violation in end point registers
Result: Legal placement solution with minimized early negative slack
1 endpoints = queryCriticalTimingEndpoints();
2 for $endpoint \in endpoints$ do
3     if isValid(endpoint) then
4         net = getClockNet(endpoint);
5         lcb = getLcb(net);
6         moveCell(endpoint, lcb.pos(), NEAR); /* Call to Algorithm 16 */
7     end
8 end
```

cell of the local clock network. In Line 5, LCB is obtained. The target optimized-timing position is the position of LCB. In Algorithm 16, the end point register is attempted to be moved closer to the target position (Line 6).

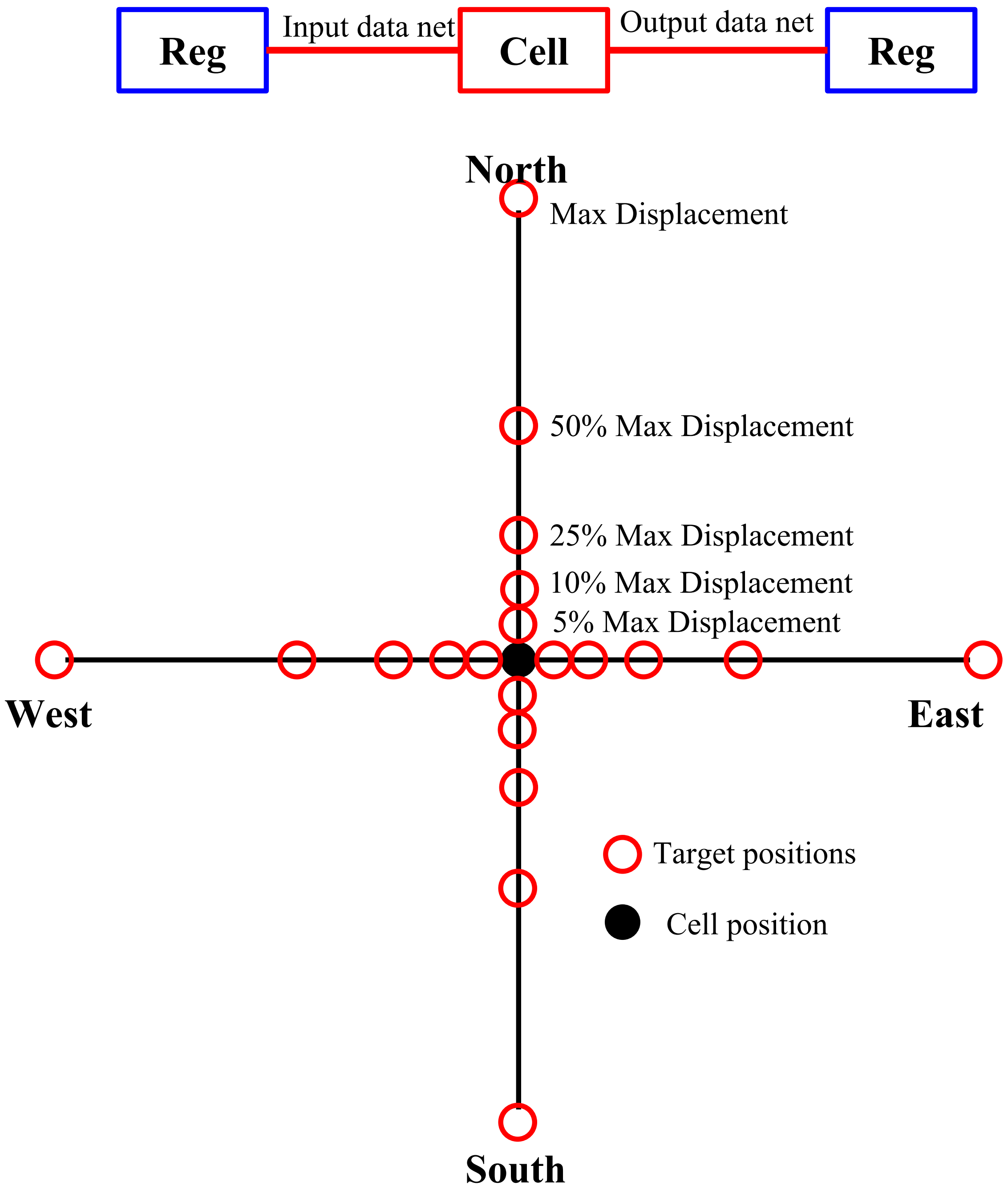
5.2.2.2 Iterative Cell Spreading

In the *Iterative Cell Spreading* algorithm, combinatorial cells with early timing violation are tentatively moved away from the initial position to increase data signal delay. These combinatorial cells are moved in the four directions (North, South, East, West) from the current position. The target position is computed over Cartesian axes with increasing step length of $\{5, 10, 25, 50, 100\}$ percent of the maximum cell displacement constraint. The algorithm places the cell in the nearest position of the optimized-timing position.

In Figure 5.5, an example of the iterative cell spreading algorithm is presented. The combinatorial cell (red) is in a path with early timing violation. This combinatorial cell can be moved away from its driver and sink cells (blue registers). In this approach, wire capacitance and resistance are increased in the input and output nets of the combinatorial cell. Therefore, data signal propagation is delayed. The best timing improvement position is searched in North, East, South, and West directions. In this figure, the black circle is the current cell position, and red circles are the target positions.

In Algorithm 6, pseudo code of the iterative cell spreading method is presented. In (FLACH et al., 2016), this algorithm is presented only to optimize timing violation. In this work, the objective is to optimize timing violations subject to routability. Therefore, optimized-timing cell positions which are computed by this algorithm are also subject

Figure 5.5: Early timing violation is minimized by increasing wire capacitance and resistance in the combinatorial critical timing cell



Source: Author (2019).

to RO restriction. This algorithm receives a legal placement solution with combinatorial cells that have early timing violation. The algorithm output is a legal placement solution with minimized early timing violation. The cell spreading algorithm computes optimized-timing positions to combinatorial cells with early timing violation.

All circuit cells are iterated (Line 1). Only valid cells (Line 2) are stored in the list (Line 3). A cell is valid only if 1) it is a combinatorial cell, 2) it has negative early

Algorithm 6: Iterative Cell Spreading

```
Data: Legal placement solution with early timing violation
Result: Legal placement solution with minimized early timing violation
1  for $cell \in cells$ do
2      if isValid(cell) then
3          list.insert(cell);
4      end
5  end
6  for $cell \in list$ do
7      bestCost = computeCostPosition(cell, cell.pos());
8      bestPos = INVALID;
9      currentPos = cell.pos();
10     for $step \in \{0.05, 0.1, 0.25, 0.5, 1.0\}$ do
11         for $direction \in \{N, E, S, W\}$ do
12             targetPos = computeTargetPos(currentPos, step, direction);
13             success = moveCell(cell, targetPos, NEAR);      /* Call to
                Algorithm 16 */
14             if sucess then
15                 cost = computeCost(cell, targetPos);
16                 if $cost < bestCost$ then
17                     bestCost = cost;
18                     bestPos = targetPos;
19                 end
20             end
21             moveCell(cell, currentPos, EXACT);
22         end
23     end
24     if $bestPos! = INVALID$ then
25         moveCell(cell, bestPos, EXACT);  /* Call to Algorithm 16
            */
26     end
27 end
```

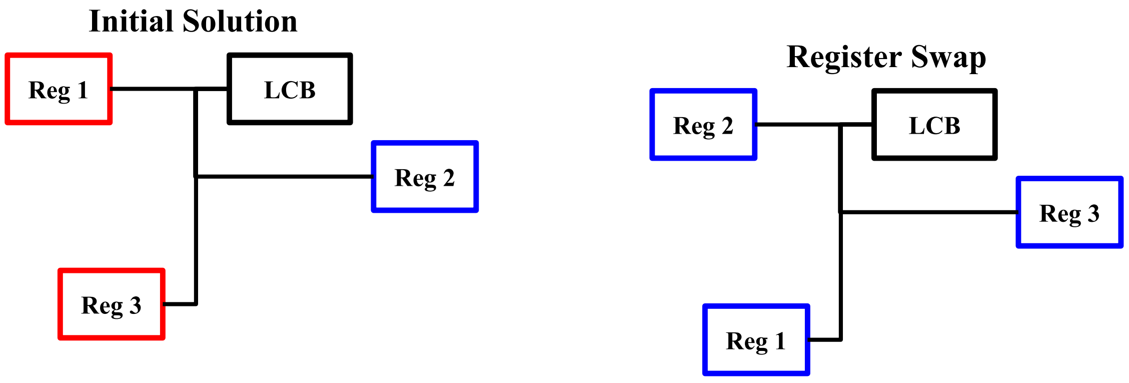
slack, and 3) it is a movable cell. All cells in the list are visited (Line 6). The best cost is initialized with the cost of the current cell position (Line 7). The best position is initialized as an invalid position (Line 8). In Line 9, the current cell position is initialized. Target position steps and cell movement directions are iterated in Lines 10 and 11, respectively. The target position is computed from the current position in terms of displacement step of the maximum cell displacement and direction of cell movement (Line 12). The critical timing cell is attempted to be moved to the target position (Line 13) using Algorithm 16. If the *move cell* algorithm can successfully move the cell to the closest position of the target position (Line 14), then the cost is computed (Line 15). The cost is a pair composed of QoR and the difference of RO between current and target positions. If the timing QoR and RO costs are improved (Line 16), then best cost and best position is updated in Lines 17 and 18, respectively. The cell is moved back to the initial position (Line 21). If a valid position is found (Line 24), then the cell is moved to the best cost position (Line 25) using Algorithm 16.

5.2.2.3 Register Swap

In the *Register Swap* algorithm, the objective is to swap registers which have common LCB to minimize early timing violations with useful clock skew. This swap algorithm also minimizes timing adverse side effects of the aforementioned iterative cell spreading algorithm. Registers are assumed to be equals. Therefore, the clock tree and clock timing characteristics are assumed not to change when registers are swapped. Moreover, the latency in each end point of the clock tree can be considered as constant. Register swap is modeled as an assignment problem similar to the problem introduced in (HELD; SCHORR, 2014). Register assignment can be obtained in polynomial time with the Hungarian algorithm (KUHN, 1955). All register positions are assumed to be available positions to assign a register.

In Figure 5.6, an example of the register swap method is presented. Register swap is computed using the Hungarian algorithm. In the new positions, registers have different clock latency compared to the previous positions. Clock latency impacts on the time of the launch and capture clock transitions. Therefore, early timing violation is minimized by optimizing clock latency in the register of critical timing paths.

Figure 5.6: Registers that have the same local clock network are swapped to minimize early timing violation



Source: Author (2019).

In Algorithm 7, pseudo code of the register swap method is presented. In (FLACH et al., 2016), this algorithm is presented only to optimize timing violation. In this work, the objective is to optimize timing violations subject to routability. Therefore, optimized-timing cell positions which are computed by this algorithm are also subject to RO restriction. The register swap algorithm receives a legal placement solution with registers that have paths with early timing violation. This algorithm returns a legal placement solution with optimized early timing violation. Register swap positions are computed using the Hungarian algorithm.

All LCBs with data paths which have early timing violation are visited (Line 1). The local clock network of the LCB is obtained (Line 2). Registers connected to the local clock network are visited (Line 3). These registers are inserted in the list (Line 4). It is assumed that registers have the same width. Each register location is available to swap with another register. In Line 6, the number of slots are obtained. A matrix $n \times n$ is built (Line 7), where n is equal to the number of slots. The register in the list index i is mapped to the matrix column i and row i. Source and target matrix indexes are the position of the respective registers in the list. The objective is to compute the lowest cost to place associated register with the source index in the location of the associated register with the target index. In Lines 8 and 9, all source and target indexes are visited, respectively. Associated registers with the source and target indexes are obtained from the list in Lines 10 and 11. Cell displacement of the source register (Line 12) is computed using the initial register position and the position of the target register. The initial cell position is the cell position when the circuit is loaded from the file. If the cell displacement of the source register is higher than the maximum cell displacement constraint (Line 13), then the cost to move the source register to the position of the target register is infinite (Line 14). The algorithm continues to iterate the next source-target pair of registers (Line 15). On the other hand, the cell displacement of the source register to the target position is lower than the maximum cell displacement constraint. The clock pin of the target register is obtained (Line 17). In Lines 18 and 19, early and late ATs at the clock pin are obtained. These ATs are early and late latency of the clock network associated with the clock pin of the target register. In 2015 ICCAD contest benchmarks, the clock network which connects the circuit clock pin to the input pins of LCBs is ideal. Therefore, the signal delay of the ideal clock network is zero. The criticality of the data pin from the source register is obtained (Line 21). In Section 3.6.5, the computation of criticality has been introduced. The cost to move the source register to the position of the target register is equal to $criticality_{dataPin} \times lateAT_{clkPin} - critcality_{dataPin} \times earlyAT_{clkPin}$ (Line 22). The cost is assigned to row and column indexes of the source and target registers. In Line 26, optimized positions of

Algorithm 7: Register Swap

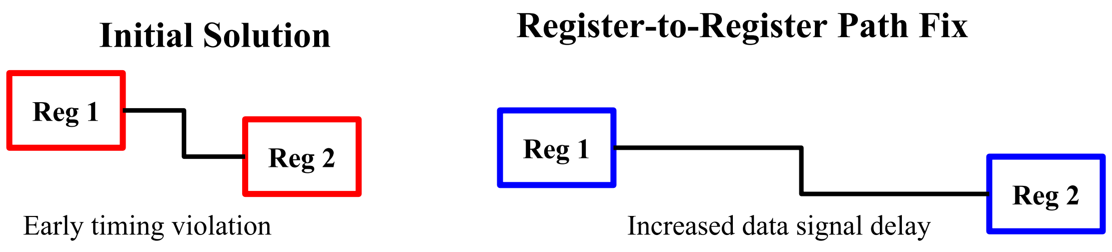
```
   Data: Legal placement solution with early timing violation
   Result: Optimized timing placement solution by swapping registers
1  for $lcb \in lcbs$ do
2      clockNet = getLocalClockNet(lcb);
3      for $reg \in clockNet$ do
4          list.insert(reg);
5      end
6      numSlots = list.size();
7      matrix.build(numSlots, numSlots);
8      for $source \leq numSlots$ do
9          for $target \leq numSlots$ do
10             regSrc = list[source];
11             regTarget = list[target];
12             disp = computeDisplacement(regSrc, regTarget.pos());
13             if $disp > maxDisp$ then
14                 matrix(source, target) = $\infty$;
15                 continue;
16             end
17             clkPin = regTarget.clockPin();
18             earlyAT = getEarlyAT(clkPin);
19             lateAT = getLateAT(clkPin);
20             dataPin = getDataPin(regSrc);
21             criticality = getEarlyCriticality(dataPin);
22             cost = (criticality $\times$ lateAT) - (criticality $\times$ earlyAT);
23             matrix(source, target) = cost
24         end
25     end
26     HungarianOptimization(matrix);
27     for $source \leq numSlots$ do
28         reg = list[source];
29         for $target \leq numSlots$ do
30             if $matrix(source, target) == 0$ then
31                 pos = list[target].pos();
32                 moveCell(reg, pos, EXACT) ;  /* Call to Algorithm
                    16 */
33                 break;
34             end
35         end
36     end
37 end
```

registers are computed using the Hungarian algorithm as introduced in (KUHN, 1955). All source indexes are iterated (Line 27). The source register is obtained from the source index in the list (Line 28). All target indexes are iterated (Line 29). The target position is the position of the target register when the first pair of source-target indexes is equal to zero (Line 30). The target position is obtained from the register associated with the target index (Line 31). The register which is associated with the source index is moved to the target position (Line 32). Algorithm iteration in the target index is broken (Line 33). This target index is the optimized-position to move the register.

5.2.2.4 Register-to-Register Path Fix

In the *Register-to-Register Path Fix*, the objective is to compute analytically shift displacement to move away from the end point register from the start point register. Moving apart from the end point register increases wire capacitance and resistance, which increases signal delay that minimizes early timing violation. A common source of early negative slacks are paths composed of two registers directly connected. These paths have no combinational cells. The early negative slack can be further improved by computing the optimized-timing position for the end point register. The path delay is composed mainly of the wire delay. In Figure 5.7, the example of the register-to-register path fix algorithm is presented.

Figure 5.7: End point register of early timing violation paths are moved away from the data driver cell



Source: Author (2019).

In Algorithm 8, pseudo code of the register-to-register path fix method is presented. In (FLACH et al., 2016), this algorithm is presented only to optimize timing violation. In this work, the objective is to optimize timing violations subject to routability. Therefore, optimized-timing cell positions which are computed by this algorithm are also subject to RO restriction. End point registers of early critical timing paths are moved away from the data driver cell. The target cell position is computed using an analytical formulation. In this approach, the driver strength of the driver cell, wire capacitance, and wire resistance are used to compute the optimized cell position. The Register-to-register path fix algorithm receives a legal placement solution with registers that have early timing violation. The output of this algorithm is a legal placement solution with minimized early timing violation.

```
Algorithm 8: Register-to-Register Path Fix
    Data: Legal placement solution with early timing violation
    Result: Optimized timing placement solution by moving away registers in
            critical timing paths
 1  paths = queryEarlyCriticalTimingPaths();
 2  for $path \in paths$ do
 3      end = path.endpoint();
 4      driver = end.getDataDriver();
 5      if $isValid(end)$ $and$ $isValid(driver)$ then
 6          disp = min(maxDisp, computeOptoDisp(end, driver));
 7          targetPos = computeTargetPos(end, disp);
 8          moveCell(end, targetPos, NEAR) ; /* Call to Algorithm 16
             */
 9          visited.insert(end);
10          visited.insert(driver);
11      end
12  end
```

Early critical timing paths are queried (Line 1). Only early critical timing paths with movable end point registers are queried. All critical timing paths are visited (Line 2). The end point register is obtained from the path (Line 3). The data driver cell is obtained from the end point register (Line 4). The end point and driver cells are valid only if they are sequential cells, and these cells were not visited previously (Line 5). Cell displacement of registers is the minimum length between maximum cell displacement and optimized cell displacement (Line 6). Optimized cell displacement to minimize early timing violation is computed as introduced in (FLACH et al., 2016). The target position is computed (Line 7). The end point register is moved near the target position (Line 8). In Lines 9 and 10, the end point register and data driver cell are inserted in the visited list, respectively.

5.2.3 Late Optimization

In late RAITDP algorithms, the objective is to reduce local signal delay. Data signals arrive in end points after late RAT. In critical timing paths, the delay of data signals is reduced by balancing wire capacitance and resistance of the input and output nets based on driver strength of the driver and moved cells. Critical timing cells are moved toward the driver or sink to change wire capacitance and resistance in input and output cells' nets.

The direction of cell movement depends on the driver strength and net characteristics. The late RAITDP flow is composed of a set of four algorithms: 1) *Clustered Cell Movement* (5.2.3.1), 2) *Buffer Balancing* (5.2.3.2), 3) *Cell Balancing* (5.2.3.3), and 4) *Driver Load Capacitance Optimization* (5.2.3.4).

5.2.3.1 Clustered Cell Movement

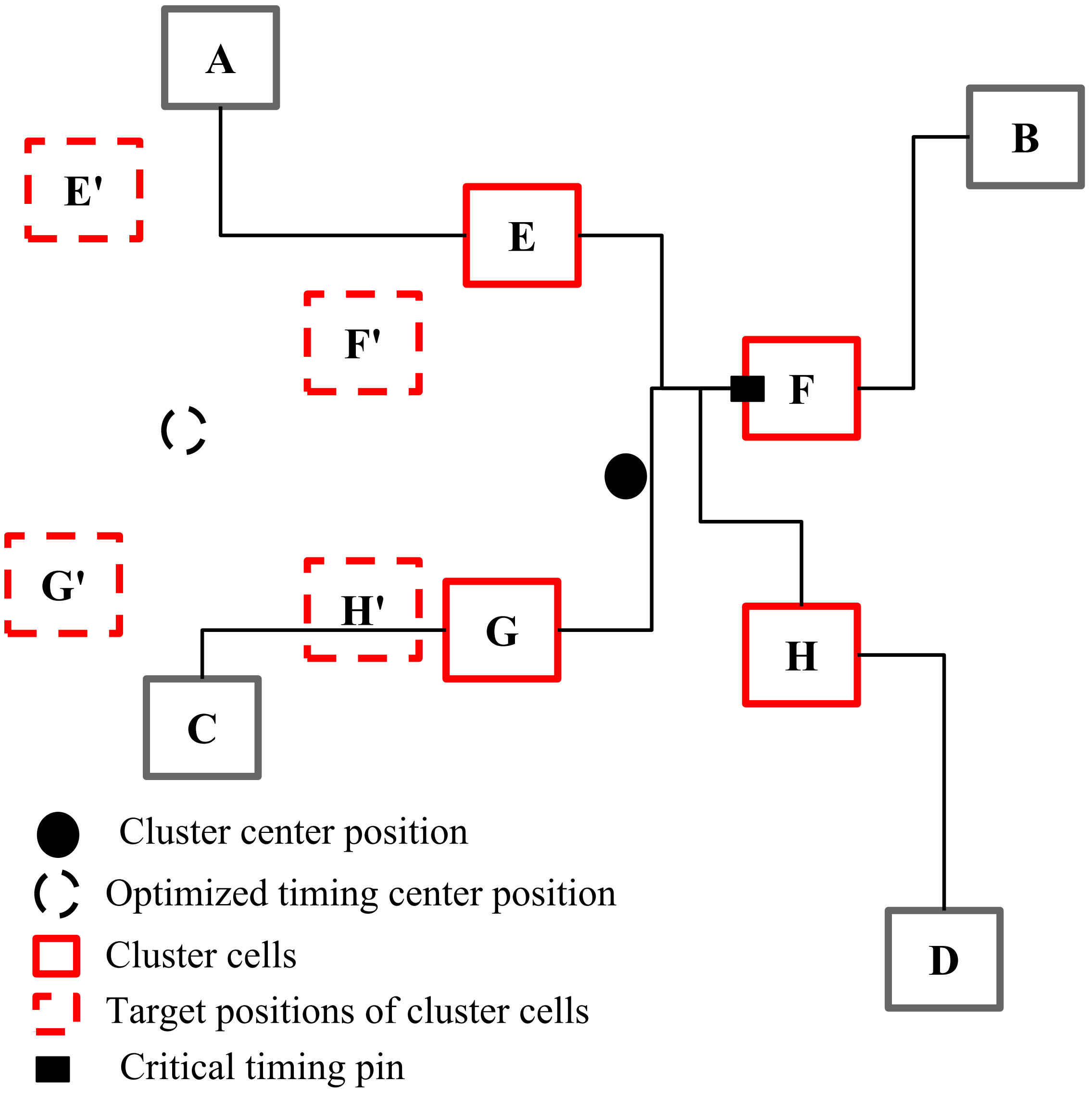
In the *Clustered Cell Movement*, a cluster of critical timing cells are moved towards the center of mass of the optimized timing position. Moving one cell at a time may lead to a suboptimal solution (BOCK et al., 2015). In this algorithm, a cluster of critical neighboring cells is built. This cluster is built from a given critical timing cell. Cells in the cluster are topological neighbors of the critical timing cell.

In Figure 5.8, an example of the clustered cell movement is presented. The cell cluster is built in topological order from a pin which has late timing violation. The center position of the cluster is computed (black circle). This position is the average position of cluster cell positions. The target cluster center position is a weighted position of late negative slack (dashed black circle) of pins from clustered cells. Cluster cells are moved toward the optimized-timing position. Each cluster cell is shifted by the difference between the center and the target positions. Cell shift is subject to the maximum cell displacement constraint.

In Algorithm 9, pseudo code of the clustered cell movement method is presented. In (FLACH et al., 2016), this algorithm is presented only to optimize timing violation. In this work, the objective is to optimize timing violations subject to routability. Therefore, optimized-timing cell positions, which are computed by this algorithm, are also subject to RO restriction. In this algorithm, neighboring topological cells from late critical timing pins are clustered. These cells are moved towards the cluster center optimized timing position. This algorithm receives a legal placement solution with late negative slack violations. The output is a legal placement solution with minimized late negative timing violation.

All combinatorial cells are visited (Line 1). Pins of the cell are visited (Line 2). Only pins with late negative slack (Line 3) are inserted in the list (Line 4). In the list, pins are sorted from the lowest to the highest negative slack (Line 8). Each pin in the list is visited (Line 9). The cell which is associated with the current pin is retrieved (Line 10). Cell cluster procedure is performed only in cells which were not moved in an early iteration of the cell cluster movement method (Lines 11 and 12). In Line 14, cells

Figure 5.8: Cluster of late critical timing cells is built. Critical timing cells are moved towards optimized-timing positions. In this approach, late timing violation is minimized



Source: Author (2019).

are clustered in topological order from the cell associated with the critical timing pin. The procedure for clustering cells is introduced in Algorithm 10. In Line 15, the center position of cluster cells and the optimized-timing center position are computed. Clustered cells are moved toward the optimized-timing position. In Algorithm 11, the procedure to compute the center position of the cluster, the optimized-timing position and to move cells toward the optimized-timing position is introduced. Moved cells are inserted in the visited list (Line 16).

In Algorithm 10, pseudo code of the method to cluster cells is presented. In this algorithm, neighboring topological cells from the cell of critical timing pin are clustered. This algorithm receives a late negative slack pin. The output is a cluster (set) of cells.

Algorithm 9: Clustered Cell Movement

```
   Data: Placement solution and critical timing cell
   Result: Cluster of cells that are neighboring of the critical timing cell
 1 for $cell \in cells$ do
 2     for $pin \in cell$ do
 3         if $getSlack(pin) < 0$ then
 4             list.insert(pin);
 5         end
 6     end
 7 end
 8 list.sort();
 9 for $pin \in list$ do
10     cell = pin.getCell();
11     if visited[cell] then
12         continue;
13     end
14     cells = clusterCells(pin) ;        /* Call to Algorithm 10 */
15     moveClusterCells(cells);           /* Call to Algorithm 11 */
16     visited.insert(cells);
17 end
```

In Line 1, the topological depth variable is initialized. In Line 2, the net is inserted in the list within respective cell topological depth. This net is associated with the pin that has negative slack. Topological neighboring cells are clustered until the list has nets and the topological depth is lower than ten (10) cells (Line 3). The topological depth was experimentally determined considering the trade-off between runtime and QoR. A pair of net and topological depth is obtained from the list (Line 4). In Lines 5 and 6, net and topological depth value are retrieved from the pair, respectively. Associated pins with the net are visited (Line 7). Associated cell with the pin of the net is obtained (Line 8). This procedure does not continue if the cell has displacement higher than ten times the row height (Lines 9 and 10). This cell displacement threshold avoids to cluster cells with massive cell displacement. Only movable cells (Line 12) are inserted in the cell cluster (Line 13). Associated pins with the cell are visited (Line 15). Slack of the visited pin is obtained (Line 16). Associated net with this pin is retrieved (Line 17). Nets of neighboring cells are inserted in the list only if 1) the net is not associated with a pin with positive slack, 2) nets are not the same and 3) net has not been visited yet (Lines 18 and 19). Associated net with this pin is inserted in the visited list (Line 21). This net and the incremented topological depth are inserted in the list (Line 22). This algorithm continues to iterate while there are nets in the list. Finally, the cluster of critical timing cells is

Algorithm 10: Cluster topological neighboring cells from a late negative slack pin.

```
   Data: Late Critical Timing Pin
   Result: Cluster of cells
 1 depth = 0;
 2 list.insert(pin.getNet(), depth);
 3 while $list.size() > 0$ and $depth < 10$ do
 4     pair = list.pop();
 5     net = pair.first();
 6     depth = pair.second();
 7     for $pinNet \in net$ do
 8         cell = pinNet.getCell();
 9         if $computeDisp(cell) > (10 \times rowHeight)$ then
10             continue;
11         end
12         if $cell.isMovable()$ then
13             cells.insert(cell);
14         end
15         for $pinCell \in cell$ do
16             slack = getSlack(pinCell);
17             netCell = pinCell.getNet();
18             if $slack \geq 0$ or pinCell == pinNet or netVisited[netCell] then
19                 continue;
20             end
21             netVisited.insert(netCell);
22             list.insert(netCell, depth+1);
23         end
24     end
25 end
26 return cells;
```

returned (Line 26).

In Algorithm 11, pseudo code of the method to compute cluster cell center position, optimized-timing position and to move cells toward the optimized-timing position is presented. The cluster center position is the average position of all cells in the cluster. The optimized-timing position is a weighted position of all pins of cells in the cluster and the pin negative slack. The distance between both cluster center and optimized-timing positions is the displacement in which cluster cells must be moved toward the optimized-timing position. The algorithm receives a cluster of cells and the legal placement solution. The output is a legal placement solution which late critical timing cells are moved to minimize timing violation.

In Line 1, all cells in the cluster are visited. Clustered cell positions are accumu-

Algorithm 11: Move Clustered Cells

```
   Data: Cluster of Cells and legal placement solution
   Result: Moved cells towards optimized timing position
 1 for $cell \in cells$ do
 2     cellPos += cell.pos();
 3     for $pin \in cell$ do
 4         net = pin.getNet();
 5         for $neighbor \in net.pins()$ do
 6             slack = neighbor.getSlack();
 7             nbCell = neighbor.getCell();
 8             if $slack > 0$ or $cell.has(nbCell)$ then
 9                 continue;
10             end
11             totalSlack += slack;
12             pinPos += $slack \times nbCell.pos()$;
13         end
14     end
15 end
16 pinCenter = pinPos / totalSlack;
17 cellCenter = cellPos / cells.size();
18 disp = pinCenter - cellCenter;
19 for $cell \in cells$ do
20     if cell.isRegister() then
21         continue;
22     end
23     targetPos = cell.pos() + disp;
24     moveCell(cell, targetPos, NEAR) ;     /* Call to Algorithm 16
        */
25 end
```

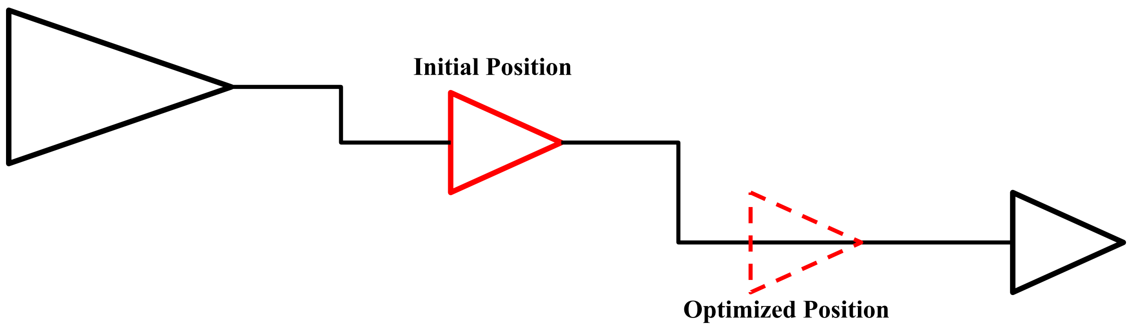
lated in the *cellPos* variable (Line 2). Pins of the cell are visited (Line 3). Associated net with the pin is obtained (Line 4). All pins in the net are visited (Line 5). Late slack and cell of the net pin are retrieved (Lines 6 and 7, respectively). If the slack is positive or the cell is in the cluster (Line 8), then the procedure goes to iterate the next pin of the net (Line 9). Late negative slack is accumulated in the variable *totalSlack* (Line 11). The multiplication of the cell position with slack is accumulated in the *pinPos* variable (Line 12). In Lines 16 and 17, cluster center and weighted-timing positions are computed, respectively. The displacement to move cluster cells is the difference between these two positions (Line 18). Critical timing cells are moved toward the weighted-timing position by the displacement distance. In Line 19, all cells in cluster are again visited to move these cells toward the optimized-timing position. Registers are not moved to avoid adverse side effects on the clock network and remaining data paths (Lines 20 and 21). The target cell position is computed (Line 23). The critical timing cell is moved closer to the target position (Line 24) using Algorithm 16.

5.2.3.2 Buffer Balancing

In the *Buffer Balancing* method, optimized-timing positions are computed in the tuple driver-buffer-sink cells to minimize signal delay locally. Optimized-timing positions are computed using an analytical equation based on the Elmore model. Usually, the circuit contains several paths with buffer chains. The required time to load output capacitance is modeled with a representative drive strength value. Optimized buffer shifting is computed considering the pin resistance of driver and buffer cells. Buffer shift is subject to maximum cell displacement constraint. The buffer must also be connected only to one driver and one sink. Both driver and sink cells are considered fixed cells.

In Figure 5.9, an example of the buffer balancing method is presented. The optimized position for the critical timing buffer (red cell) is computed. Moving the critical timing cell (dashed triangle) closer to its sink cell minimizes local signal delay in the output net of the critical timing cell. However, the signal delay is increased in the input net. In this case, the driver cell has a higher driver strength than the critical timing cell. Therefore, the sum of the signal delay in the input and output nets of the critical timing buffer may be reduced. The time to load capacitance in large cells is less affected by capacitance changing when comparing the time to load the capacitance by small cells.

Figure 5.9: Critical timing buffer is moved to optimized position. In this position, the total signal delay is locally minimized in input and output nets of the buffer



Source: Author (2019).

In Algorithm 12, pseudo code of the method to compute and to move the buffer to the local optimized-timing position is presented. In (FLACH et al., 2016), this algorithm is presented only to optimize timing violation. In this work, the objective is to optimize timing violations subject to routability. Therefore, optimized-timing cell positions which are computed by this algorithm are also subject to RO restriction. Pin capacitance of the driver and the buffer cells and wire capacitance and resistance of the input and output nets of the buffer are used to compute the optimized-timing position. The algorithm receives a legal placement solution with buffers that have late timing violation. The output is a legal placement solution with minimized late timing violation.

Algorithm 12: Buffer Balancing

```
Data: Legal placement solution
Result: Legal placement solution with minimized late timing violation
1  for $cell \in cells$ do
2      slack = cell.getSlack();
3      if $cell.isBuffer()\ and\ slack < 0$ then
4          list.insert(cell);
5      end
6  end
7  list.sort();
8  for $buffer \in list$ do
9      driver = buffer.getDriver();
10     sink = bufffer.getSink();
11     pos = computePos(driver, buffer, sink);
12     moveCell(cell, targetPos, NEAR) ;     /* Call to Algorithm 16
        */
13 end
```

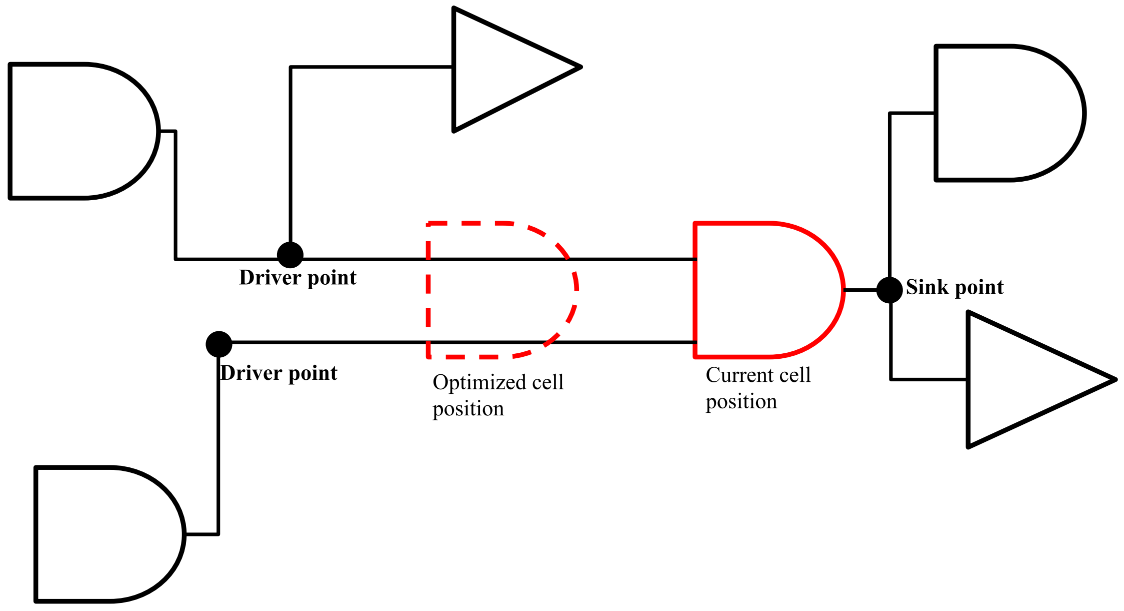
All circuit cells are visited (Line 1). The slack of the cell is obtained (Line 2). Only buffers with negative slack (Line 3) are inserted in the list (Line 4). The buffer's driver must have only one sink, which is the buffer, and the buffer must be connected to only one sink. Otherwise, the buffer balancing algorithm is not useful to minimize timing violation. In the list, buffers are sorted from the highest to the lowest criticality (Line 7). Criticality is introduced in Section 3.6.5. In the sorted list, buffers are visited (Line 8). In Lines 9 and 10, the driver and sink cells from the buffer are obtained, respectively. Optimized timing position to place the buffer is computed as introduced in (FLACH et al., 2016) (Line 11). Finally, the buffer is moved closer to the optimized-timing position (Line 12) using Algorithm 16.

5.2.3.3 Cell Balancing

In the *Cell Balancing*, the same approach to computing optimized-timing positions to buffers is extended to cells. Cells may have several input pins. Moreover, cells may have several nets connected to the input and output pins. Each of these nets may have several cells. In this approach, driver and sink positions of the cell are weighted positions of cells in input and output nets. An optimized-timing position is computed between input and output equivalent positions. The optimized-timing position is computed using an analytical formulation. The analytic formulation of cell balancing is modeled in the same fashion as the buffer balancing formulation.

In Figure 5.10, an example of the cell balancing method is presented. In the cell balancing algorithm, optimized timing positions are computed for critical timing cells, which may have input and output nets with several cells. The source and sink points may be Steiner tree points of the input and output nets or the position of driver cells in input nets and sink cells in output net. The optimized timing position is computed in terms of driver strength of the drivers from input nets and the driver strength of the critical timing cell. In the optimized timing position, the summation of the signal delay in the worst timing path passing through the critical timing cell is reduced.

Figure 5.10: Late critical timing cells are moved to optimized-timing positions



Source: Author (2019).

In Algorithm 13, pseudo code of the method to move late critical timing cells to optimized-timing positions is presented. In (FLACH et al., 2016), this algorithm is presented only to optimize timing violation. In this work, the objective is to optimize timing violations subject to routability. Therefore, optimized-timing cell positions which are computed by this algorithm are also subject to RO restriction. The optimized-timing position is a weight position considering driver and sink cells in input and output nets of the critical timing cell. This algorithm has two versions. In the first version, driver and sink positions are positions of driver and sink cells of input and output nets, respectively. In the second version, driver and sink positions are positions of Steiner points of input and output nets. The algorithm input is a legal placement solution with cells that have late timing violation. The output is a legal placement solution where the timing violation is minimized in late critical timing paths.

Algorithm 13: Cell Balancing

```
Data: Legal placement solution with cells that have late timing violation
Result: Legal placement solution with minimized late timing violation
for $cell \in cells$ do
    if $isValid(cell)$ then
        list.insert(cell);
    end
end
list.sort();
totalPos = 0;
totalWeight = 0;
for $cell \in list$ do
    for $outPin \in cell.outPins()$ do
        outNet = outPin.getNet();
        for $inPin \in cell.inputPins()$ do
            inNet = inPin.getNet();
            driver = inNet.getDriverPin();
            wDriver = $\frac{2 \times centrality(driver) + criticality(driver)}{3}$;
            for $sink \in outNet.getSinkPins()$ do
                wSink = $\frac{2 \times centrality(sink) + criticality(sink)}{3}$;
                pos = computeTargetPos(cell, inPin, sink);
                totalPos += (wSink + wDriver) * pos;
                totalWeight += wSink + wDriver;
            end
        end
    end
    targetPos = totalPos / totalWeight;
    moveCell(cell, targetPos, NEAR) ;      /* Call to Algorithm 16
      */
end
```

All circuit cells are visited (Line 1). Only valid cells (Line 2) are inserted in the list (Line 3). A cell is valid if it is movable and it has late negative slack. Registers and LCBs are not inserted in the list. In Line 6, cells are sorted from the highest to the lowest gain on timing violation improvement. Cell gain is a weighted mean value to estimate the timing violation improvement that can be obtained in a cell. In (5.1), the equation to estimate timing improvement gain for a cell is presented.

$$gain(cell) = \frac{1}{A}\sum_{pin}^{pin \in cell}\sum_{TA} DR \times OC \times \frac{2 \times centrality(pin) + criticality(pin)}{3} \quad (5.1)$$

where A is the number of timing arcs in all output pins of the cell. pin is an output pin of the cell. TA is a timing arc associate with an output pin. DR is the driver resistance. OC is the output load capacitance associated with the output pin. Centrality and criticality are computed as introduced in Section 3.6.5.

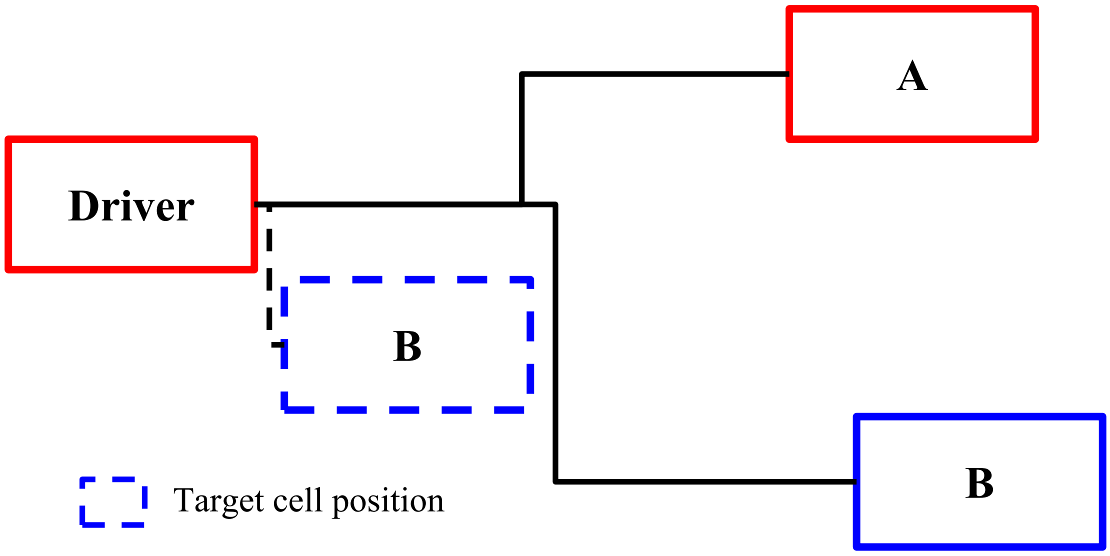
Variables to accumulate the summation of positions and weights are initialized in Lines 7 and 8, respectively. In the list, all critical timing cells are visited (Line 9). In this stage, all cell paths composed of the driver-cell-sink tuple are listed. For each path, a weighted timing position is computed. All output pins associated with the cell are visited (Line 10). Net associated with the output pin is obtained (Line 11). All input pins associated with the cell are visited (Line 12). The input net associated with the input pin is retrieved (Line 13). The driver pin of the input net is obtained (Line 14). The timing weight of the driver pin is computed (Line 15). All sink pins associated with the output net are visited (Line 16). The timing weight of the sink pin is computed (Line 17). Target position of the path driver-cell-sink is computed as introduced in (FLACH et al., 2016) (Line 18). In Lines 19 and 20, total weight positions and total timing weights are accumulated, respectively. The target position is the weighted timing mean position for all driver-cell-sink paths (Line 24). The cell is moved closer to the target position (Line 25) using Algorithm 16.

5.2.3.4 Driver Load Capacitance Optimization

In the *Driver Load Capacitance Optimization* method, the objective is to reduce the capacitance in the non-critical timing branches of the critical timing nets. In critical nets with more than two cells, sink cells with positive slack may be moved closer to their driver cells. Therefore, the required time to load the output capacitance (wire capacitance) is reduced. This approach directly minimizes local timing delay. Each cell movement is evaluated if the moved cell is free of timing violation. Cell delay is updated locally and incrementally. If a new timing violation is created, then cell movements are undone.

In Figure 5.11, an example of the driver load capacitance optimization method is presented. In critical timing nets, non-critical timing cells are moved closer to their driver cells. In this approach, wire capacitance is reduced by minimizing wire length in critical timing nets. Therefore, the time to load output capacitance (wire and port capacitance) is reduced, and the signal delay is minimized. In the example, non-critical timing *cell B* is moved closer (dashed space) to its driver cell. Therefore, the wire length to connect *cell B* to the net is reduced.

Figure 5.11: In late critical timing nets, non critical timing sink cells are moved closer to their driver cells. In this approach, wire capacitance of branches is reduced in critical timing nets



Source: Author (2019).

In Algorithm 14, pseudo code of the method to move non-critical timing sink cells near their critical timing driver cells is presented. In (FLACH et al., 2016), this algorithm is presented only to optimize timing violation. In this work, the objective is to optimize timing violations subject to routability. Therefore, optimized-timing cell positions which are computed by this algorithm are also subject to RO restriction. The algorithm receives a legal placement solution with cells that have late critical timing violation. The output is a legal placement solution that late critical timing violation is minimized.

All circuit nets are visited (Line 1). Only nets which have the driver pin with negative late slack (Line 2) are inserted in the list (Line 3). In the list, nets are sorted from the highest to the lowest cost (Line 6). The net cost is computed as follows: $critically(net) \times wireCapacitance(net)$. Criticality is introduced in Section 3.6.5. In the list, all nets are visited (Line 7). Associated cell driver with the net is obtained (Line 8). Sink pins of the net are visited (Line 9). Sink cells which are fixed or that have late negative slack are not moved closer to their driver cells (Lines 10 and 11). Target positions are computed for cells that may be moved closer to their driver cells (Line 13). This

```
Algorithm 14: Driver Load Capacitance Optimization
Data: Legal placement solution with late timing violation
Result: Legal placement solution with minimized late timing violation
for $net \in nets$ do
    if $getSlack(net) < 0$ then
        list.insert(net);
    end
end
list.sort();
for $net \in list$ do
    driver = net.getDriver();
    for $sink \in net$ do
        if $sink.getSlack() < 0$ or $sink.isFixed()$ then
            continue;
        end
        pos = computeTargetPos(driver, sink);
        moveCell(sink.getcell(), pos, NEAR) ; /* Call to Algorithm 16 */
    end
end
```

method has two versions: 1) target position is the position of the driver cell in the critical timing net and 2) target position is the position of Steiner point of the wire segment connected to the input pin of the sink cell. The target position is subject to the maximum cell displacement constraint. The sink cell is moved closer to the target position (Line 14) as presented in Algorithm 16.

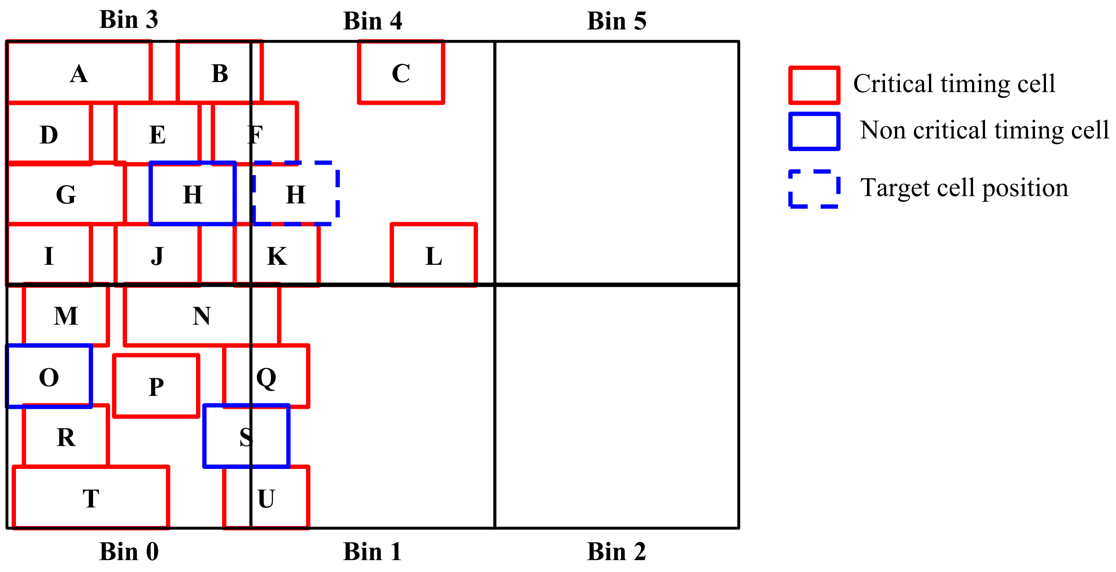
5.2.4 Area Density Optimization

In the *Area Density Optimization* method, the objective is to minimize area violations in high-density regions. Late incremental timing-driven placement algorithms tend to accumulate critical timing cells in small regions. This cell concentration may cause violation in area density utilization. The area density optimization is a simplified cell spreading algorithm. Cells are moved out from high-density regions subject to late negative slack, maximum cell displacement constraint, and routability.

The area density utilization is evaluated with the ABU metric. ABU is introduced in Section 3.6.2. The placement area is modeled as a graph of regular bins. A bin has area density violation if the total cell area divided by bin capacity is higher than the maximum area density constraint. In the area density optimization algorithm, overfilled bins are flagged. In each flagged bin, the algorithm searches if a neighboring bin has white space to receive cells. Cells are ranked by the lowest to highest late slack in flagged bins. Cells with the highest positive slack are moved first. Only cells with positive late slack can be moved. One cell at a time is moved to a neighbor bin with the free area while the neighbor bin has space and bin remains with area overfill. Cell slack is incrementally updated. Cell movements which cause cell timing violations are undone.

In Figure 5.12, an example of the area density optimization method is presented. In the example, non critical timing *cell H* is moved out from *bin 3* to *bin 4*. In *bin 3*, the area density violation is minimized by moving out *cell H*.

Figure 5.12: Moving non critical late timing cells from overfilled bins to a neighboring bin with white space. In this approach, area density violation is minimized by moving out non critical timing cell



Source: Author (2019).

In Algorithm 15, pseudo code of the method to move cells out from high-density regions is presented. In (FLACH et al., 2016), this algorithm is presented only to optimize timing violation. In this work, the objective is to optimize area density and timing violations subject to routability. Therefore, optimized-timing cell positions which are computed by this algorithm are also subject to RO restriction. Only non-critical late timing cells are moved out from overfilled bins. The algorithm searches only in neighboring bins from the cell bin with area density violation. This algorithm receives a legal placement solution with area density violation. The output is a legal placement solution with minimized area density violation.

All circuit cells are visited (Line 1). for each bin with area density violation, only

Algorithm 15: Area Density Optimization

```
Data: Legal placement solution with area density violation
Result: Legal placement solution with area density optimization
for $cell \in cells$ do
    if $cell.getSlack() > 0$ $and$ $hasAreaViolation(cell)$ then
        list.insert(cell);
    end
end
for $cell \in list$ do
    bin = getNeighboringBin(cell);
    if $bin! = INVALID$ then
        pos = computeTargetPosition(bin, cell);
        moveCell(cell, pos, NEAR) ;   /* Call to Algorithm 16 */
    end
end
```

cells that have positive late timing slack (Line 2) are inserted in the list (Line 3). In the list, all cells are visited (Line 6). In Line 7, the neighboring bin which is free of area density violation is returned. Otherwise, this procedure returns an invalid reference to indicate that there is no neighboring bin with the available area to receive the cell. In this procedure, only the first topological level of neighboring bins is searched. The target position is computed (Line 9), only for valid bins (Line 8) . In the target position, the cell must be entirely placed inside of the neighboring bin. The cell is moved to the target position (Line 10) as presented in Algorithm 16.

5.2.5 Cell Movement Algorithm

In Algorithm 16, the pseudo code of the method to move cells to target positions or near target positions is presented. In (FLACH et al., 2016), this algorithm is presented only to optimize timing violation. In this work, the objective is to compute optimized-timing positions subject to routability and cell displacement restrictions. This algorithm operates in two modes. In the first mode, cells are placed in target positions. In the second mode, legal white spaces near the target positions are searched to place cells. In some incremental timing-driven techniques, the exact position to place cells is known. Therefore, it is not necessary to search white spaces again for these optimized-positions. The algorithm receives the cell to be placed, the target position and the mode ($mode \in \{NEAR, EXACT\}$). The output is the cell placed in the target position or near to the target position and an indication that cell movement was executed. The algorithm also indicates that it is not possible to move the cell closer to the target position.

Algorithm 16: Move Cell Algorithm

```
    Data: Cell, Target Position and $mode \in \{NEAR, EXACT\}$
    Result: Fail or placed cell in target position or in the nearest position from
            the target position
 1  if mode == EXACT then
 2      placeCell(cell, pos);
 3      return true;
 4  end
 5  if mode == NEAR then
 6      for $step \in \{5, 10, 25, 50, 100\}$ do
 7          width = $step \times cell.width()$;
 8          region = computeSearchRegion(cell, targetPos, width, maxDisp);
 9          pos = findNearestWhitespacePosition(cell, targetPos, region);
10          if isValid(pos) and $checkRoutingOF(cell.pos(), pos)$ then
11              placeCell(cell, pos);
12              return true;
13          end
14      end
15  end
16  return false;
```

In Line 1, the cell movement mode is checked if it is exact mode. In *exact mode*, the cell is placed in the target position (Line 2) and the algorithm returns indicating that the cell was moved to the target position (Line 3). In exact mode, the *move cell* algorithm assumes that the target position is a legal and valid position in terms of routability and maximum cell displacement constraints.

In Line 5, the cell movement mode is checked if it is *near mode*. In this mode, a legal white space to move the cell is searched from the target position up to five restricted search regions (Line 6). The maximum width of the search area is computed (Line 7). The search region is determined using the cell, target position, the cell width and maximum cell displacement constraint (Line 8). The region boundaries are limited by maximum cell displacement and maximum region width. In Line 9, the nearest white space location from the target position is searched in the region. This search procedure returns a valid position if it has found a white space. Otherwise, this search procedure returns an invalid position. The position is checked if it is valid. In this position, the RO in the target position is verified if it is lower than the RO in the current cell position (Line 10). Cell movement can be accepted if the current and target cell positions are inside of the same

GCell. If the cell movement is valid, the cell is placed in the target position (Line 11) Therefore, this algorithm returns true to indicate the cell has been successfully moved to the target position (Line 12). The default return of the cell move algorithm is false, to indicate that the cell has not been successfully moved to the target position (Line 16).

5.3 Experimental Results

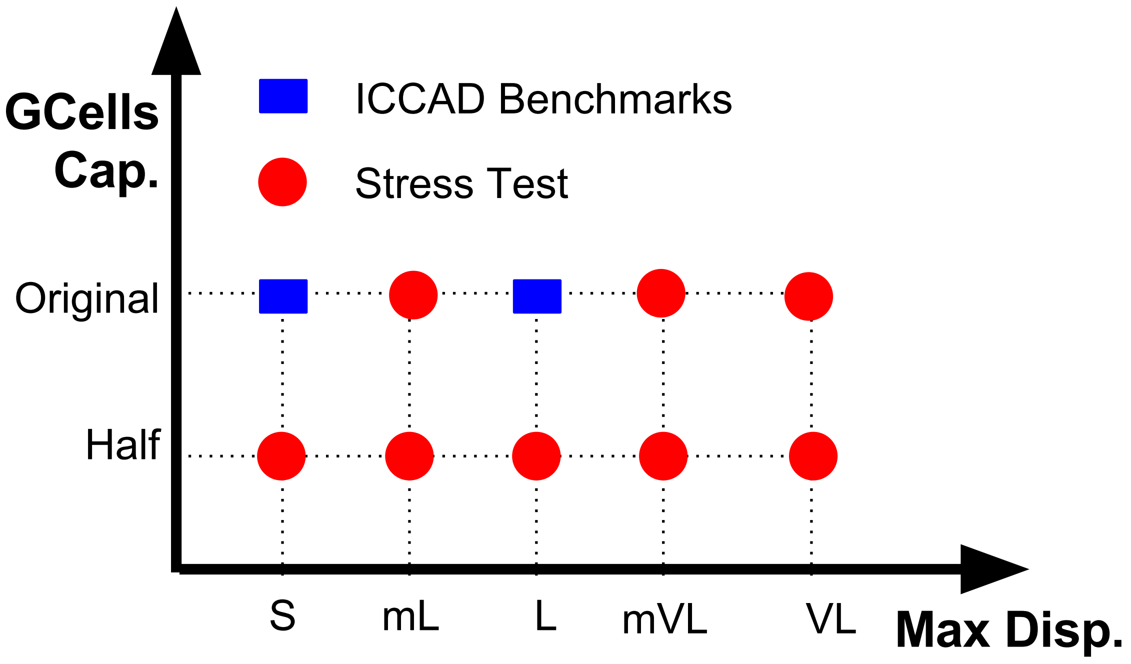
In this section, experimental setup and numerical results are presented. First, the circuit delay and RO improvements are discussed. The algorithm has been evaluated with ten corner cases. In each corner case, cell displacement constraint and routing resources are changed. Experimental results are compared to the top three teams from the 2015 ICCAD contest (KIM et al., 2015) and (FLACH et al., 2016) solutions.

5.3.1 Experimental Setup

The proposed RAITDP algorithm has been developed in C++-11 and compiled with GCC 4.8.3. Experimental evaluations are conducted on a CentOS 6.5 server with two Intel Xeon E5-2620 processors running at 2.00 GHz and 64 GB of RAM. All experiments have been performed on 2015 ICCAD benchmarks (KIM et al., 2015). The RO is evaluated using the global router NCTUgr (LIU et al., 2013). The circuit delay is statically measured using the Golden Timer (HUANG; WONG, 2015) from the 2015 ICCAD contest (KIM et al., 2015). The timing propagation in nets connected to moved cells is incrementally updated after each cell movement using a built-in timer. The RO is updated for the entire circuit after each iteration of timing optimization algorithms, as NCTUgr only routes the entire circuit. It has no feature to reroute a small set of nets incrementally.

A comprehensive set of experiments are conducted to analyze different aspects of RO and timing violations improvement on the proposed algorithm. The maximum cell displacement constraint and the wire width and spacing of routing layers are modified from the 2015 ICCAD benchmarks. This approach changes the search space to move a cell and the routing resources. The cell displacement constraint limits the maximum cell movement, which impacts on the improvement of timing violations. Wire width and spacing are modified to reduce the GCell capacity. The routing constraint becomes more challenging to achieve when the routing resources are reduced. In Figure 5.13, the experimental configuration to evaluate the proposed algorithms is presented. In the ordinate, two GCell capacities are presented. In the abscissa, five maximum cell displacement constraints are addressed. The entire evaluation setup is composed of ten corner cases. Two corner cases are taken from the 2015 ICCAD benchmarks. In the remaining corner cases, the GCell capacity and the cell displacement constraints are changed from the 2015 ICCAD benchmarks.

Figure 5.13: Cell displacement constraint and GCell capacity corner cases to comprehensively evaluated timing violations and RO improvement from the ICCAD benchmarks



Source: (MONTEIRO et al., 2016)

The *Original* GCell capacity is the available routing resources from the 2015 ICCAD benchmarks. In this case, the wire spacing and width are kept the same as defined in the contest benchmarks. In the *Half* GCell capacity, the available routing resources in GCells are reduced by half. The wire width and spacing are increased twice from the original value. This approach reduces the available routing resources by half, which leads to a challenging routability circuit while optimizing timing delay. The cell displacement constraints are adjusted based on short (S) and long (L) restrictions which have been established in the 2015 ICCAD benchmarks. The medium large (mL), medium very large (mVL) and very Large (VL) displacement constraints are computed as shown in Equations 5.2, 5.3, and 5.4, respectively.

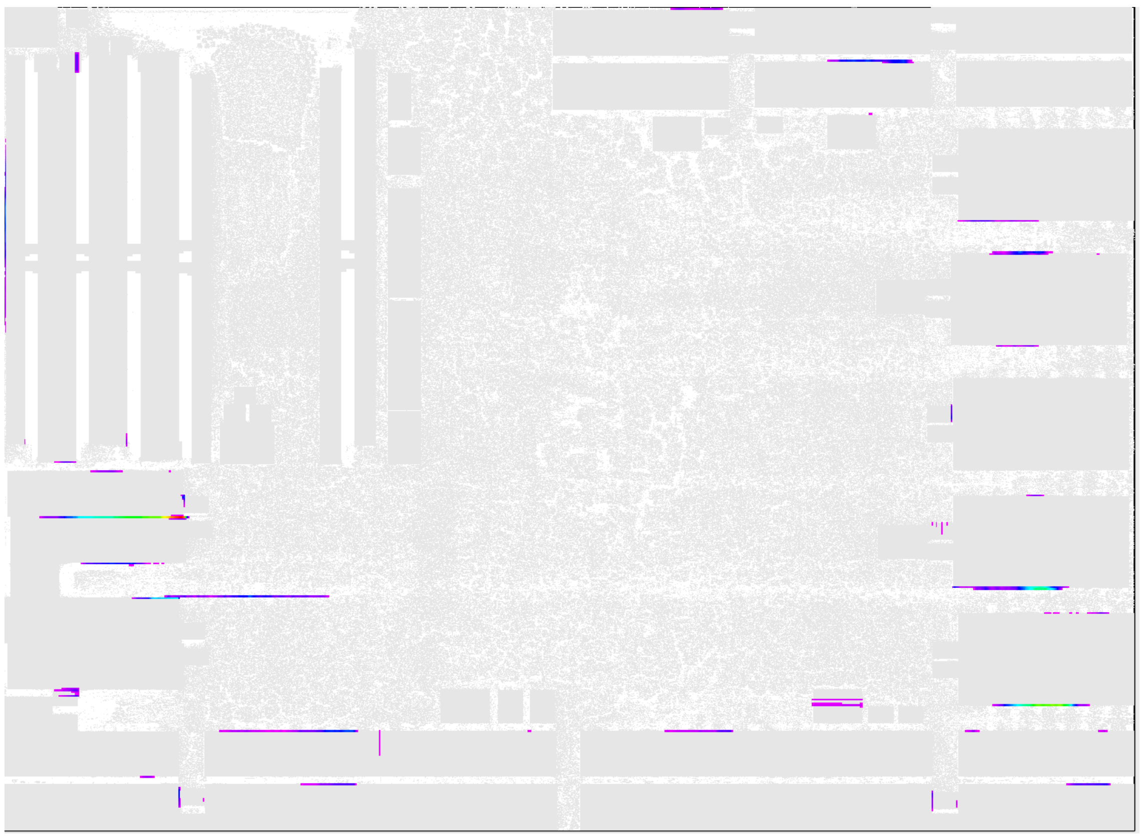
$$mL = \frac{S+L}{2} \tag{5.2}$$

$$mVL = \frac{3 \times L}{2} \tag{5.3}$$

$$VL = 2 \times L \tag{5.4}$$

In the 2015 ICCAD benchmarks, there is no significant RO. The main RO is primarily around the macroblocks. Therefore, the proposed optimization algorithm is hard to evaluate adequately. In Figures 5.14 and 5.15, examples of RO in *original* and *half* GCell capacities for Superblue16 are presented. The RO violations are the colorized regions. When the routing resources are reduced by half, the RO has increased significantly. In this case, the proposed RAITDP algorithm can be adequately evaluated.

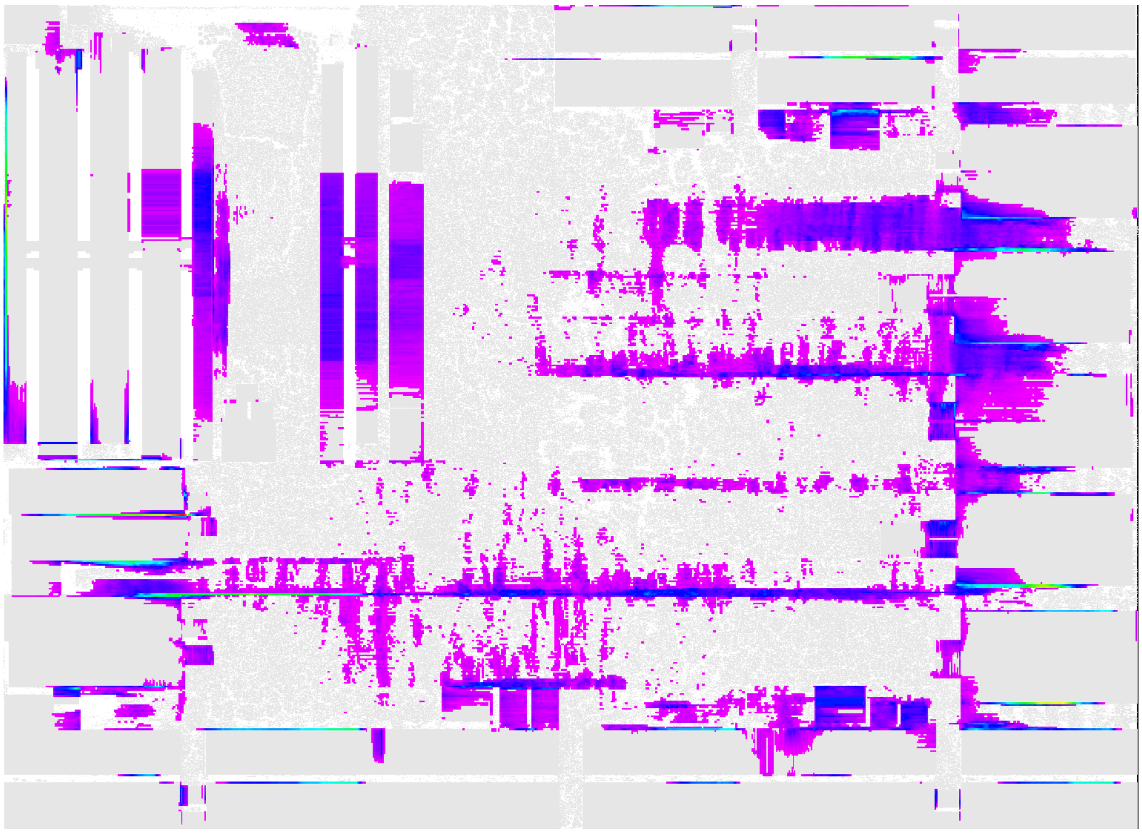
Figure 5.14: In the original GCell capacity, there is no significant RO from the 2015 ICCAD benchmarks



Source: (MONTEIRO et al., 2016)

The improvement on timing and routing violations are compared to the top three teams from the 2015 ICCAD contest and (FLACH et al., 2016). The RO from the final solution of the ICCAD contest teams are evaluated with NCTUgr. The improvement of timing violation is obtained from the contest award presentation. Timing and routing results for (FLACH et al., 2016) are obtained by running the binary and evaluating the

Figure 5.15: In the half GCell capacity, there is more RO to properly evaluate the proposed RAITDP



Source: (MONTEIRO et al., 2016)

RO of the final solution.

5.3.2 Numerical Results

In Table 5.1, the timing violations and RO improvements are presented. The experimental results from the proposed RAITDP algorithm are compared to the top three teams from the 2015 ICCAD contest and (FLACH et al., 2016). Timing and routing results are analyzed only for *(Original, S)* and *(Original, L)* corners. In Table 5.1, columns 1 and 2 indicate the benchmarks (BM) and maximum cell displacement constraints (Dis). The TQoR is presented from column 3 to 7 to the top three contest teams, ITDP, and RAITDP algorithms. The initial total Global Routing Overflow (GRO) (GP #OF) is shown in column 8. The total GRO improvement for the top three teams, ITDP and RAITDP solutions are highlighted in columns 9 to 13. The runtime, in seconds, to optimize timing in placement (ITDP and RAITDP algorithms) and to execute the global router (NCTUgr) are indicated in columns 14 to 16, respectively.

Table 5.1: Results comparing the developed RAITDP with the top 3 teams from the 2015 ICCAD contest and ITDP (FLACH et al., 2016) for the set *Original* GCell capacity of Fig. 5.13

BM	Dis	TQoR Improvement					GP #OF	#GRO Improvement					Runtime (s)		
		1st	2nd	3rd	ITDP	RAITDP		1st	2nd	3rd	ITDP	RAITDP	ITDP	RAITDP	Routing
sb1	Short	**448**	234	411	392	391	109,925	-180	-144	-1	-3,162	**-3,511**	715	695	+752
sb3		243	160	214	351	**360**	5,541	-6	-85	-41	-514	**-584**	836	840	+1,097
sb4		**288**	80	179	276	266	61,969	28	24	-11	-326	**-329**	573	541	+527
sb5		41	98	148	**149**	**149**	41,631	-59	-154	-31	-1,765	**-2,742**	685	654	+926
sb7		98	43	70	**141**	140	3,289	-165	-8	-14	**-195**	-194	845	867	+953
sb10		112	61	**146**	143	138	1,451,260	-600	-480	-170	-13,700	**-15,440**	1,202	1,210	+1,654
sb16		525	370	386	735	**736**	16,044	-200	208	-213	**-2,490**	-2,237	669	700	+738
sb18		**365**	180	258	302	302	3	**-3**	-2	**-3**	**-3**	**-3**	471	484	+569
Avg.		265	153	227	**311**	310	-	-148	-80	-61	-2,769	**-3,130**	749	749	+902
sb1	Long	347	164	0	**508**	499	109,925	-272	-287	0	-2,245	**-3,429**	508	585	+475
sb3		552	428	404	**756**	474	5,541	24	18	65	-283	**-406**	756	825	+1,084
sb4		507	209	0	**666**	598	61,969	-74	-7	0	**-1,042**	-633	666	574	+580
sb5		180	249	247	**470**	358	41,631	-771	-583	-91	-3,507	**-5,724**	470	377	+345
sb7		201	39	130	**264**	254	3,289	**-294**	-38	-31	-198	-168	264	823	+832
sb10		181	232	162	**493**	403	1,451,260	-550	10	160	-16,300	**-30,560**	493	938	+955
sb16		895	394	559	**1,209**	1,069	16,044	-972	862	-260	-3,684	**-3,787**	1,209	440	+439
sb18		613	355	484	**781**	686	3	**-3**	**-3**	**-3**	**-3**	**-3**	781	364	+354
Avg.		434	259	248	**643**	543	-	-364	-4	-20	-3,408	**-5,589**	643	616	+633

For the short cell displacement constraint, the RAITDP algorithm achieves similar TQoR compared to (FLACH et al., 2016), on average. The proposed algorithm improves GRO by 13% compared to (FLACH et al., 2016), on average. The top three teams have no significant improvement in RO. The RO restrictions can effectively help to improve routability without penalizing circuit timing. The required runtime to route the circuit is higher than the runtime of the timing-driven placement optimization algorithms. The extra runtime can significantly help to improve the quality of the placement solution. Therefore, an improved placement solution can significantly minimize runtime and routing issues in the following physical synthesis stages.

In Table 5.2, the numerical results of the proposed algorithm for the set of corner cases shown in Figure 5.13 are presented. Columns 1 and 2 indicate the benchmarks (BM) and maximum cell displacement constraints (Dis). The TQoR is presented from column 3 to 7 when the cell displacement constraint is changed. The short (S) corner case is the baseline to evaluate TQoR. The remaining TQoR is the representative timing violation improvement achieved by relaxing the cell displacement constraint. The positive values indicate the improvement in timing violations. The initial total GRO (GP #OF) is shown in column 8. The total GRO improvement by changing the cell displacement constraint is highlighted from column 9 to 13. Negative values indicate the reduction in the total violations in the global routing.

Table 5.2: Timing violation and routing overflow improvement by changing cell displacement constraint and GCell capacity as introduced in Figure 5.13.

BM	Dim.	TQoR Improvement					GP #OF	#GRO Improvement				
		S	mL	L	mVL	VL		S	mL	L	mVL	VL
sb1	Half GCell Cap.	359	+64	+80	+93	**+102**	636,961	**-4,962**	594	2,290	1,549	2,692
sb3		305	+175	+169	**+179**	+168	481,199	**-1,715**	402	1,081	768	727
sb4		251	+315	+347	+364	**+365**	448,119	**-270**	15,653	19,233	20,095	20,962
sb5		141	+116	+217	**+490**	+224	319,619	**-4,815**	-4,353	326	1,471	935
sb7		132	+103	+122	+132	**+145**	226,349	**-989**	-208	-151	140	-124
sb10		125	+257	+278	**+283**	+280	3,675,000	-10,590	**-27,940**	-25,110	-23,940	-22,670
sb16		660	+378	+409	**+470**	+457	431,425	-6,257	-15,405	-17,742	**-19,633**	-18,447
sb18		301	+307	+385	+393	**+396**	73,502	**147**	1,381	925	1,045	1,134
Avg.		284	+214	+251	**+301**	+267	-	-3,681	**-3,735**	-2,394	-2,313	-1,849
sb1	Original GCel Cap.	391	+90	+108	**+113**	+112	109,925	**-3,511**	-2,719	-3,429	-3,014	-3,070
sb3		360	+359	+423	+411	**+426**	5,541	**-584**	-488	-406	-544	-540
sb4		266	+353	+401	+426	**+442**	61,969	-329	-565	-633	-681	**-819**
sb5		149	+202	+314	+325	**+341**	41,631	-2,742	-3,717	-5,724	-5,602	**-5,810**
sb7		140	+105	+122	+136	**+141**	3,289	-194	**-202**	-168	-157	-188
sb10		138	+293	+310	**+320**	+308	1,451,260	-15,440	**-31,870**	-30,560	-30,960	-29,110
sb16		736	+405	+474	+484	**+489**	16,044	-2,237	-3,297	**-3,787**	-3,589	-3,657
sb18		302	+401	+479	+483	**+490**	3	**-3**	**-3**	**-3**	**-3**	**-3**
Avg.		310	+276	+329	+337	**+344**	-	-3,130	-5,358	**-5,589**	-5,569	-5,400

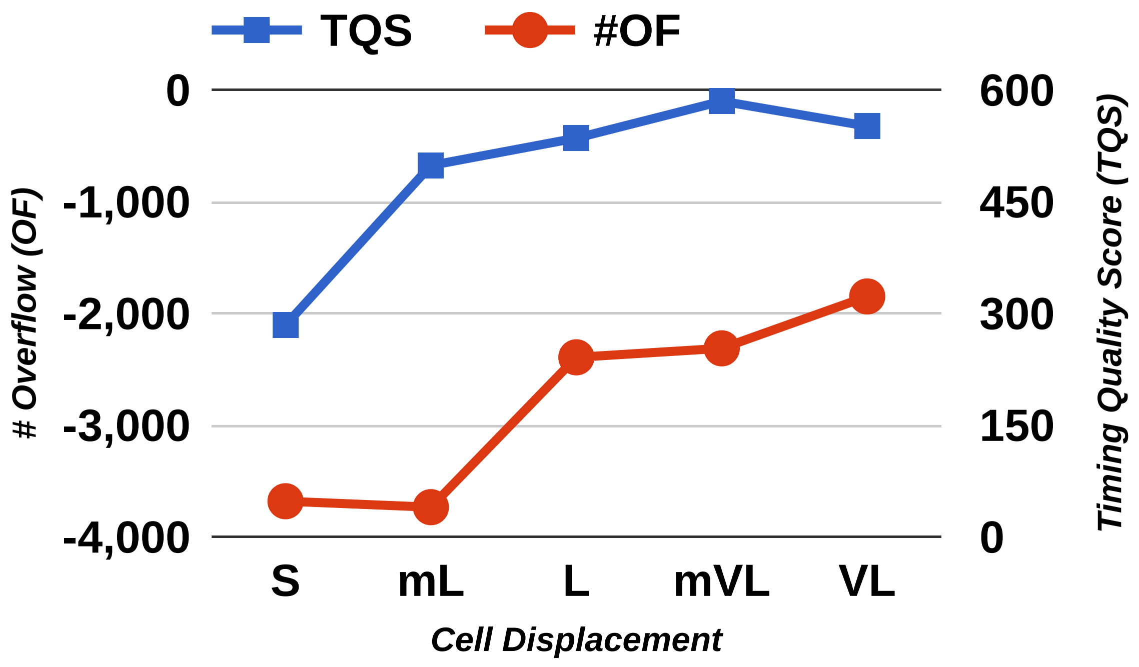
In the corner cases with half GCell capacity, the TQoR improvement increases. On the other hand, the improvement in GRO is reduced when the cell displacement constraint is less restrictive. The reduction of RO improvement is mainly due to the long cell movements, which must cross several GCells. These cell movements can cross some GCells which have RO. The global router is only able to route the entire circuit.

In the original GCell capacity corners, the timing violation, and RO are improved when the cell displacement constraint is relaxed. The primary RO improvement is obtained by moving critical timing cells away from macroblocks. In this corner case, the most critical RO is in the macroblock boundaries.

In both GCell capacities, *half* and *original*, the TQoR improves significantly in the *mL* cell displacement constraint compared to the *S* cell displacement constraint. The local optimum cell displacement for most of the critical timing cells is between these two displacement constraints. Moreover, in the half GCell capacity, there is a small RO improvement in the *mL* compared to *S* cell displacement constraints.

In Figures 5.16 and 5.17, the timing violations and RO improvement for the *half* and *original* GCell corner cases are presented. In the abscissa, the variation of cell displacement constraint is shown. In the left and right ordinates, the TQoR and GRO (#OF) are presented, respectively.

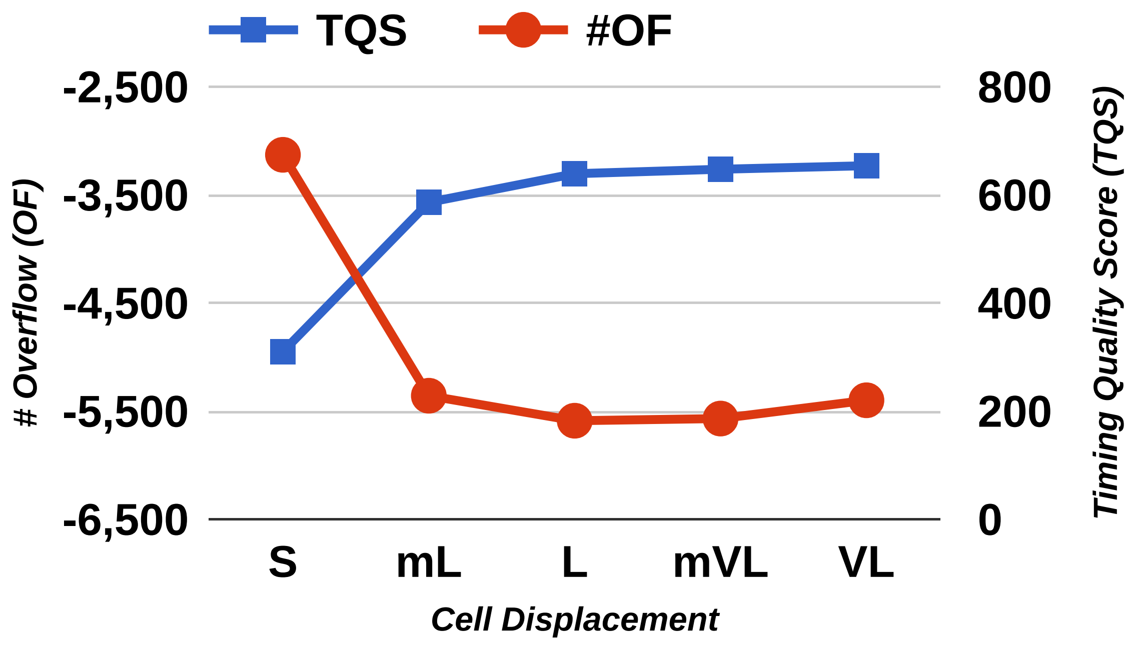
Figure 5.16: The average RO and quality scores for the set $Half$ GCell capacity



Source: (MONTEIRO et al., 2016)

In circuits that have RO near macroblocks, the proposed algorithm can signifi-

Figure 5.17: The average RO and quality scores for the set *Original* GCell capacity



Source: (MONTEIRO et al., 2016)

cantly improve timing and routability as demonstrated in Figure 5.17. In this case, the optimized-timing cell movements which move cells distant from macroblocks have a higher chance to be accepted. The target position has a higher chance to be free of RO. By moving cells distant from the RO regions, the routability violations can be significantly improved.

5.4 Summary

In this chapter, the proposed RAITDP algorithm has been presented. The objective is to optimize the circuit delay subject to routability. Optimized-local positions are computed for critical timing cells. The optimized-timing position is evaluated if it is free of RO. A cell is moved to the optimized position only if both timing and routability are improved. The improvement of routability and timing is analyzed by changing the cell displacement constraint and routing resources. The proposed algorithm improves timing and routability in circuits that have soft and hard constraints in terms of timing and routing.

In detailed placement, cell spreading can be a tricky optimization. Cells can be directly moved to a position in a region with available space. However, this procedure can have adverse side effects on moved cells. For example, some cells which did not have timing violation can have timing violation in the target position. Another adverse side effect is the significant increase in wire length of cell's nets when the cell is moved far away. A better approach would be to move the cell to an intermediate and closed position. In the intermediate position, a chosen cell is moved to another intermediate position. This procedure can be repeated while a moved cell does not reach a region free of constraint violation. In this procedure, a flow (set of cells) is moved, and a path of cells associated with target positions is created.

The second benefit of this approach is to open space in high-density regions with minimal adverse side effects on moved cells. Optimization algorithms can explore these spaces in detailed placement, which can find better positions. Therefore, the quality of the placement solution can be further improved. A cell spreading algorithm with minimized adverse side effects on moved cells can further improve the QoR in ITDP and RAITDP algorithms. Moreover, this procedure could be inserted at the beginning of the timing-driven placement optimization flow to open space in high-density regions. These opened spaces could be used by the ITDP algorithms to further improve timing violations.

6 THE PROPOSED NETWORK FLOW-BASED CELL SPREADING ALGORITHM

6.1 Introduction

In this chapter, the proposed network flow-based cell spreading algorithm is presented. This algorithm is applied in legalization and detailed placement optimization. Application and experimental results of the proposed cell spreading algorithm in legalization and detailed placement are given in Chapters 7 and 8, respectively.

The proposed cell spreading algorithm is a hybrid technique. Branch and cut and network flow techniques are combined. The hybrid cell spreading algorithm provides an optimized placement solution which may not be obtained using separated techniques.

The objective of the proposed cell spreading algorithm is to move cells out from high-density regions. Cells are moved with minimized cell displacement cost. Optimized-cost paths are computed to move sets of cells between neighboring intermediate bins. The proposed cell spreading algorithm searches optimized-cost paths in n-ary trees. The root bin of each tree is an overfilled bin. Neighboring bins are inserted in a priority queue. In the priority queue, the bin with the lowest cell displacement cost is opened first. Neighboring bins are inserted in the children list of the opened bin. Therefore, the branch of the search tree increases one level. Each branch of the tree is a candidate path. Tree branches which have higher cell displacement cost than the upper limit bound are pruned. The upper limit cost is the cell displacement cost of the minimum cost path. Every time a new path with lower cost is found, then the minimum cost path is replaced. Therefore, the upper limit cost is reduced. In each algorithm iteration, an n-ary tree is built for each overfilled bin.

A cell displacement cost model, which provides the direction and history of cell movement is integrated into the proposed cell spreading algorithm. Cell movements are subject to cell displacement constraint. Other restrictions (e.g., timing, and routability) may be included in the proposed cell spreading algorithm. Several cells can be moved with reduced runtime and minimized adverse side effects during area density optimization. Spaces in high-density regions are available with minimized cell displacement cost. These newly available spaces can then be used in legalization and detailed placement to further improve wire length, cell displacement, timing, power, and routability.

The main contributions of the proposed cell spreading algorithm can be summarized as follows:

- A cell spreading algorithm that integrates network flow and branch and cut techniques.
- A max-flow min-cost technique to compute optimized-cost paths to spread cells from high-density regions.
- New constraints can be easily integrated into the cost function.
- Using a cell cost displacement model which indicates the direction and history of cell movement to compute cell displacement cost.
- Cell displacement is optimized by exploring the direction and history of cell movements.
- A network flow-based cell spreading algorithm which can be used in legalization and detailed placement stages.

This chapter is organized as follows: In Section 6.2, the overview, and insights of the proposed algorithm are presented. In Section 6.3, the grid graph of the proposed cell spreading algorithm is addressed. In Section 6.4, the proposed network flow-based cell spreading algorithm is given. In Section 6.5, the summary of this chapter is shown.

6.2 Overview of the Proposed Network Flow-based Cell Spreading Technique

Usually, greedy cell spreading algorithms move cells directly to low-density regions. In this approach, moved cells may have severe adverse side effects. Adverse side effects may be excessive power consumption, routability issues, timing violation, and increasing wire length. On the other hand, an amount of cell area can be moved to an intermediate region. A cell area may be moved out from the intermediate region to another region. Intermediate regions may be inside or outside high-density regions. In this second approach, adverse side effects are minimized in moved cells.

6.2.1 An Example of Cell Spreading

Placement core may be divided into a set of bins. A bin is a rectangular area which covers a small piece of the placement area. Neighboring bins are connected to establish a grid graph. In each bin, the total cell overlap area is added. A bin has area density violation (overfilled bin) when the total cell area is higher than the maximum placeable area constraint. Therefore, supply is higher than zero.

In Figure 6.1, two examples of cell spreading from an overfilled bin are presented. Initial placement solution is given in Figure 6.1a. *Bin 0* has area density violation. Therefore, one cell must be moved out to alleviate area density violation. In Figure 6.1b, a greedy cell spreading procedure is presented. Cell *Inst 4* is moved out from *Bin 0* directly to *Bin 2*. *Bin 2* has available white space to receive the incoming cell without causing or increasing area density violation. On the other hand, *Inst 4* may have severe adverse side effects because of high cell displacement.

In Figure 6.1c, an optimized cell spreading approach is presented. In this approach, cells are moved out from high-density regions to intermediate regions. In intermediate regions, the second group of cells is moved out to another intermediate region. This procedure is iterated until an intermediate region may receive incoming cells without causing or increasing area density violation.

Cell *Inst 4* may be moved out from *Bin 0* to *Bin 1*. *Bin 1* is an intermediate bin. If *Inst 4* is moved to *Bin 1*, then area density violation will be created. A cell must be moved out from *Bin 1* to open space to receive *Inst 4*. In *Bin 1*, *Inst 8* may be moved to *Bin 2*. Therefore, enough space is opened in *Bin 1* to receive *Inst 4*. This sequence of bins and flagged cells to be moved is a path. In this procedure, cells are moved out from high-density regions with reduced adverse side effects. Optimized cost paths to move cells out from high-density regions are challenging to compute. The proposed cell spreading algorithm searches optimized-cost paths based on this approach.

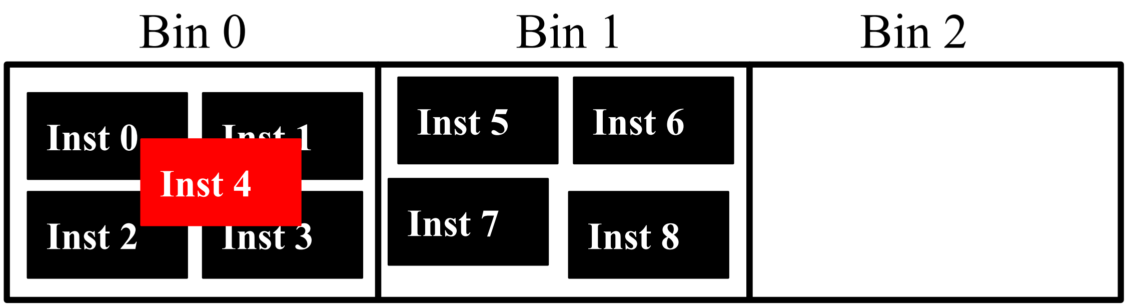
6.2.2 Overview of the N-ary Tree to Search Optimized-Cost Paths

In the proposed network flow-based cell spreading technique, cells are moved out from regions with area density violations by moving cells between neighboring bins. For each overfilled bin, a search tree is built to compute an optimized-cost path to move cells. The proposed algorithm iterates while bins have area density violation. In each algorithm iteration, a search tree is built for each overfilled bin. A new search tree is built in each algorithm iteration if the bin has area density violation.

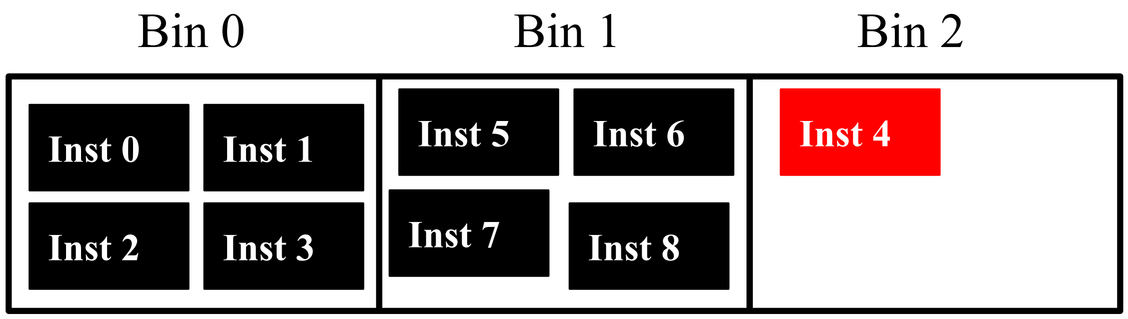
In legalization and detailed placement, the proposed network flow-based cell spreading algorithm uses non-overlapping and overlapping grid graphs, respectively. Non-overlapping and overlapping grid graphs are introduced in Section 3.7. Search trees are computed in the same fashion in grid graphs of non-overlapping and overlapping modes.

Figure 6.1: Cell spreading approaches are presented. Cells can be directly moved to the nearest white space location. Optimized-cost paths can be computed to move cells out from high-density regions through intermediate regions to a low-density region.

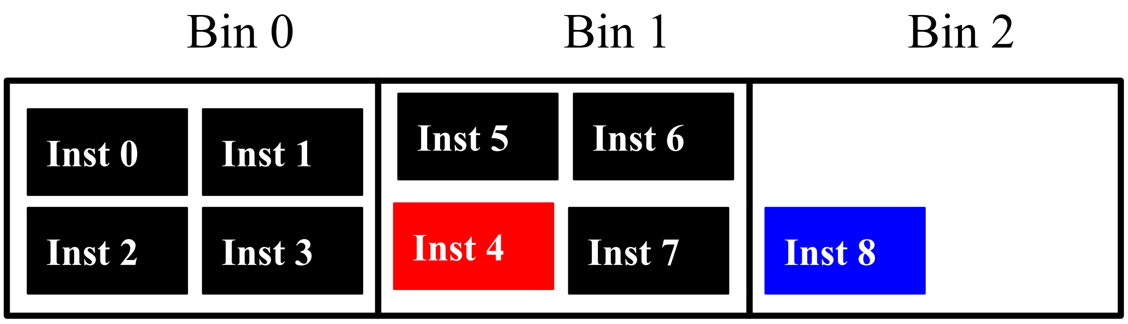
(a) Initial placement solution



(b) Move cell to the nearest white space



(c) Optimized-cost path to move cells



Source: Author (2019).

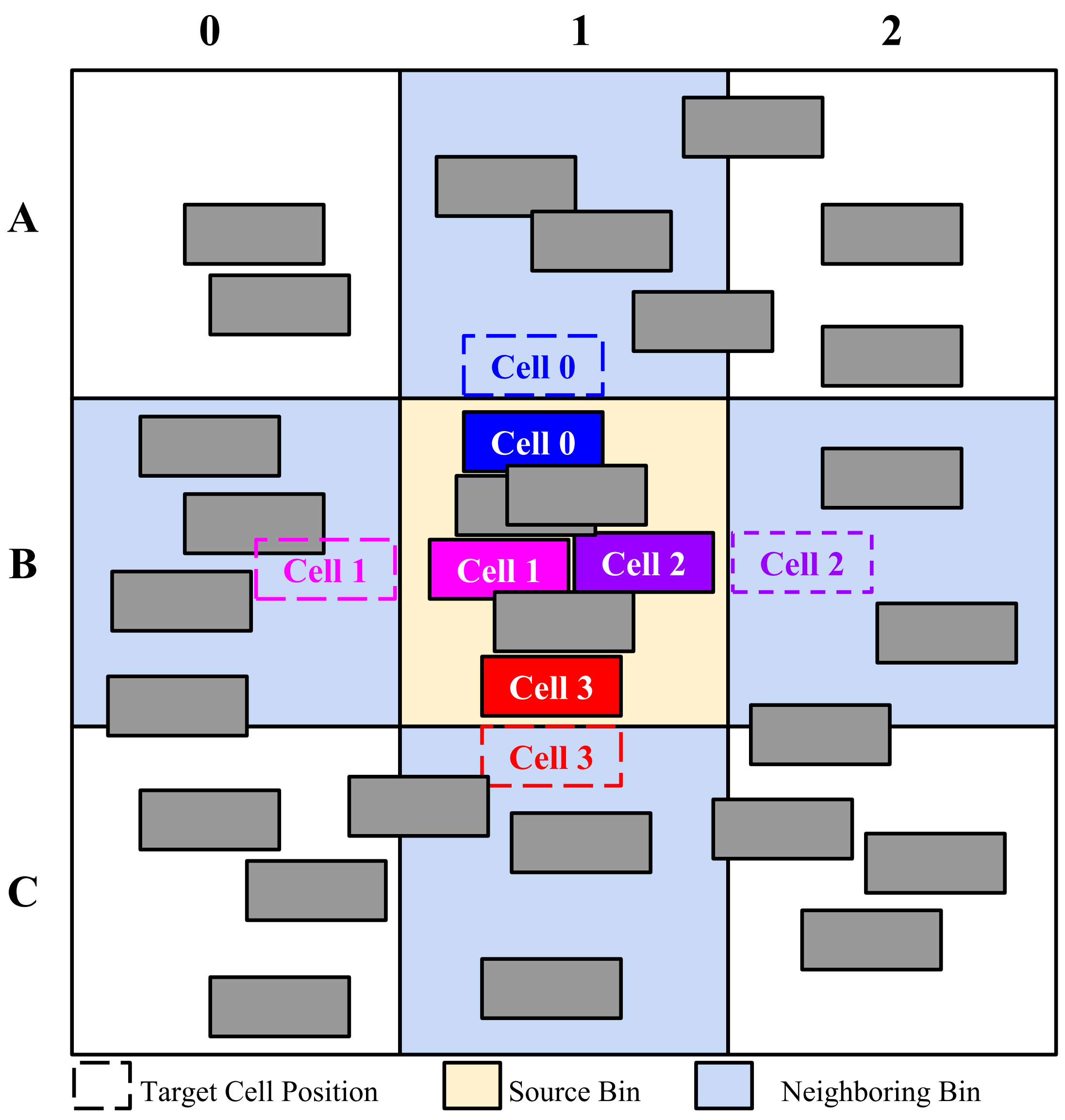
In Figure 6.2, an example of visiting neighboring bins and flagged cells are presented. Bin *B1* is an overfilled bin. Therefore, one cell must be moved out to fix area density violation. There are four candidate paths from *B1*: 1) B1-A1, 2) B1-B2, 3) B1-C1, and 4) B1-B0. For each path, there is a flagged cell which can be moved out from B1. Cells which have the lowest cost (displacement) are flagged first. For each neighboring bin, a different set of cells are flagged as presented as follows:

- *Cell 0* (blue) can be moved from B1 to A1.

- *Cell 1* (pink) can be moved from B1 to B0.
- *Cell 2* (purple) can be moved from B1 to B2.
- *Cell 3* (red) can be moved from B1 to C1.

This procedure is repeated for each bin in the search tree, which is not a leaf bin. A bin is a leaf if it has enough space to receive the inflow cell area without creating area density violation. In the search tree, if neighboring bins have not been visited before, then they are visited. Bin B1 may be the root (overfilled) bin or an intermediate bin. This procedure is used to flag cells and to compute the outflow area. Each set of flagged cells to be moved to a neighboring bin provides an outflow area of bin B1 to each neighboring bin.

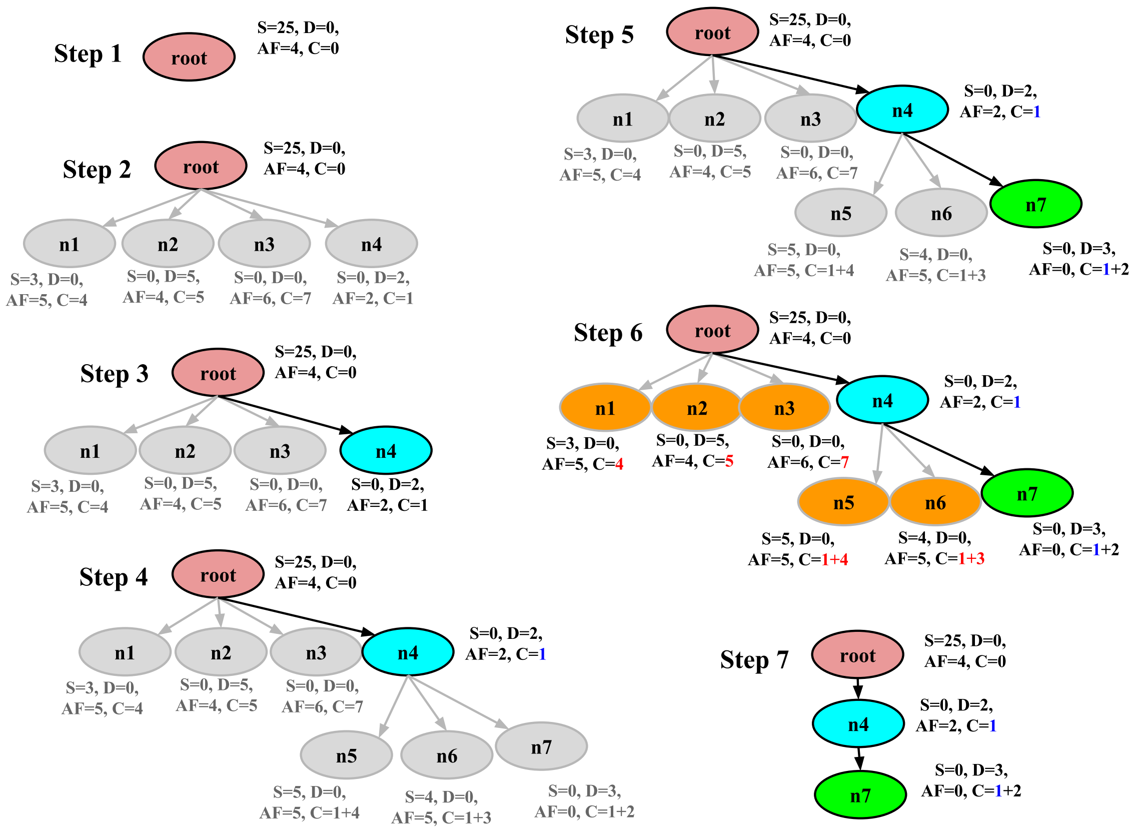
Figure 6.2: Example of flagging cells and computing outflow area



Source: Author (2019).

In Figure 6.3, an example of search tree is presented. In *Step 1* of Figure 6.3, *root* is an overfilled bin. Root bin is inserted in the priority queue. Bins are sorted by the lowest to the highest cell displacement cost of the path. In the root bin, a different set of cells are flagged to be moved out from root to each neighboring bin (an example is presented in Figure 6.2). Cell displacement cost is computed as introduced in Section 3.6.6. Bin cost to move cells from a bin to a neighboring bin is the summation of cell displacement of flagged cells. Each pair (source bin and neighboring bin) may have a different bin cost. The path cost is the summation of costs from the bin and ancestor's bins.

Figure 6.3: Example of search tree to compute optimized-cost path



Source: (MONTEIRO; JOHANN; BEHJAT, 2019)

In Figure 6.3, properties of each bin are defined as follows:

- S is the supply.
- D is the demand.
- AF is the area flow.
- C is the cost.

Supply and *demand* area spaces are the overfilled area and the available area that are evaluated to receive incoming cells. Area flow is the amount of area which is moved out from a bin to a neighboring bin. Cost is the cell displacement cost of the path.

In *Step 1*, an overfilled bin is obtained from the list of overfilled bins. This bin is the root node of the search tree. The outflow area of the root bin is computed. This computed outflow area must be moved to a neighboring bin.

In *Step 2* of Figure 6.3, the root bin is retrieved from the priority queue. All neighboring bins (n1 to n4) are visited. Sets of cells and cell displacement costs from root to each neighboring bin are computed. Each neighboring bin is inserted in the priority queue alongside its cost. In each neighboring bin, it is verified if there is enough space to receive the incoming inflow area.

In *Step 3* of Figure 6.3, the bin in priority queue with the lowest cost is retrieved. *Bin n4* has the lowest cost. Root bin has an output flow of four area units to bin n4. Bin n4 has two area units of demand. Therefore, bin n4 can absorb two area units. However, two area units must be opened in the n4 to avoid creating area density violation.

In *Step 4* of Figure 6.3, neighboring bins (n5 to n7) of bin n4 are visited. Bin n4 has a path with the lowest cell displacement cost. For each neighboring bin, a set of cells is flagged. The total cell displacement cost of this set of cells is computed. Neighboring bins of bin n4 are inserted in the priority queue.

In *Step 5* of Figure 6.3, the bin with the lowest cost is retrieved. *Bin n7* has the lowest cost. *Bin n7* has four area units of demand (available area to receive inflow area). This bin will receive two area units from *bin n4*. Therefore, *bin n7* can receive the inflow area without causing area density violation. This bin is flagged as the leaf bin of the best cost path. Remaining bins in the priority queue must be visited to verify if there is a path with lower cost.

In *Step 6* of Figure 6.3, remaining bins in the priority queue are retrieved. In this example, all remaining bins have cost higher than the cost of the best path. Therefore, these bins are not visited. Otherwise, if the bin cost is lower than the best path, this bin is visited, and neighboring bins are inserted in the priority queue.

In *Step 7* of Figure 6.3, the optimized cost path is presented. This path is composed of *root*, *n4*, and *n7* bins. Searching path algorithm returns the reference of the leaf bin (*n7*) of the minimum cost path. This algorithm may fail to find a leaf bin which can receive the inflow area without causing area density violation. In this case, this algorithm returns an invalid reference to the leaf bin.

The invalid reference indicates that the searching path algorithm has failed to find an optimized cost path for the root (overfilled) bin. In a future iteration of the cell spreading algorithm, a valid path to this overfilled bin may be found. Available cells to be flagged in intermediate bins may be different in each algorithm iteration, which may allow the path augmentation algorithm to find a valid path. Therefore, in the modified circuit placement, the cell spreading algorithm may find sets of cells to move between neighboring bins, which can find a valid path.

In this search procedure, cells are only flagged to later be moved. This procedure does not move cells between neighboring bins. In this procedure, the optimized-cost paths with flagged cells are computed. A specialized algorithm receives the optimized-cost path and moves flagged cells from parent to child bins. In Section 6.2.3, the cell movement algorithm is presented.

6.2.3 Cell Movement

In the valid optimized-cost paths, cells are moved only between neighboring bins. Paths are iterated in backtrack order. In the first iteration of the algorithm, flagged cells are moved out from *bin n4* to *bin n7*. Therefore, the required white space is opened in *bin n4*. In the end of the first iteration in *bin n4*, there are four area units available to receive the incoming cell area from *root bin*. Two area units are already available, and two area units have been opened by moving out a set of flagged cells. In the second iteration, cells are moved out from *root bin* to *bin n4*. Therefore, area density violation is minimized in the *root bin*.

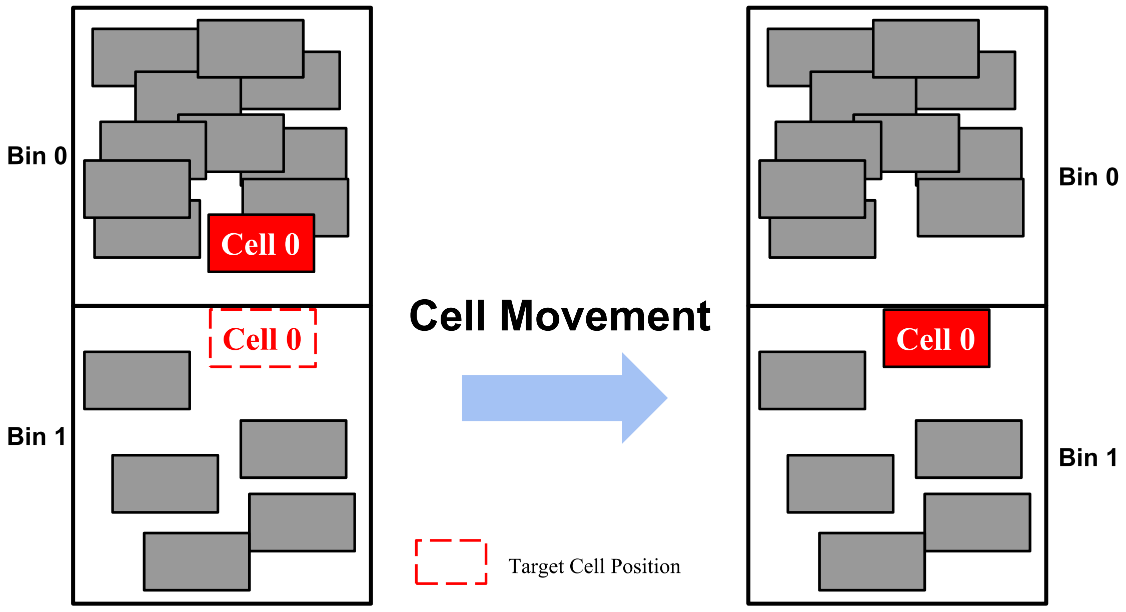
In Figure 6.4, an example of cell movement between neighboring bins is presented. In this example, *cell 0* (red) is moved out from *bin 0* to *bin 1*. Cell 0 is placed in the closed position where the cell is entirely inside the bin. The cell movement algorithm is performed in the same fashion for overlapping and non-overlapping grid graphs.

6.3 Grid Graph

In the proposed cell spreading algorithm, non-overlapping and overlapping grid graphs are used to search optimized-cost paths. These grid graphs are introduced in Section 3.7. In non-overlapping, the grid graph has no overlapping with macroblocks and fixed cells. In overlapping, bins of the grid graph may have overlap with macroblocks and fixed cells.

In legalization, the non-overlapping type of grid graphs is used. The bin height is equal to row height, and cells are aligned to rows. Moved cells may be placed only

Figure 6.4: Example of cell movement between neighboring bins



Source: Author (2019).

partially in neighboring horizontal bins. In vertical cell movements, cells must be moved entirely to the top or bottom bin.

In detailed placement, the overlapping type of grid graphs is used. The bin height is higher than the row height, and cells may not be aligned to rows. Moved cells must be placed entirely inside of bins. Cells are placed in the closest position from the current position where the cell is entirely inside of the bin.

The root node of the search tree is a bin which has area density violation. The order to visit overfilled bins has no significant impact on placement solution. An amount of cell area is computed to be moved to a neighboring bin. Neighboring bins are visited in the left, right, top, and bottom order. In each visited bin, it is evaluated if the bin has enough space to receive incoming cell area without causing or increasing area density violation. Otherwise, the algorithm searches for several cells to be moved out to open the required space to receive the incoming cell area. This procedure is iterated while the bin which has enough area to receive the incoming cell area is not found.

6.4 The Proposed Cell Spreading Algorithm

In the proposed network flow-based cell spreading algorithm, branch and cut technique and an enhanced cell displacement cost model is integrated. The enhanced cost model is introduced in Section 3.6.6. Branch and cut technique is used to prune candidate paths which have total cell displacement cost higher than the upper limit bound. In the cost model, the direction of cell movement from the initial cell position is computed. Cell movements away from initial positions have positive cost values. On the other hand, negative cost values indicate that cells are going to be moved closer to the initial positions. The same cell can belong to different paths in the same or different iterations of the cell spreading algorithm. Cells which have been moved away from their initial positions in a previous optimized-cost path can be moved closer to the initial position in a future path. In the cost model, it is easy to identify and prioritize cells, which may be moved closer to the initial position. The cost signal, positive or negative, indicates the direction of cell movements away from or closer to the initial positions, respectively. In this approach, cell displacement cost can be further optimized.

In the circuit core, a grid of bins is built. Bins may or may not have overlap with fixed cells. In Section 3.7, the two types of grid graphs (non-overlapping and overlapping) are presented. In legalization, the grid graph is built with the non-overlapping mode. In this approach, bins are small. Regions with a massive cell density are very hard to optimize area density utilization with the non-overlapping grid graph. In detailed placement, the grid graph is built with the overlapping approach. In this approach, bins are bigger compared to bins in non-overlapping mode. Therefore, it is easier to move cells out from regions which have massive area density utilization.

An optimized-cost path is a sequence of neighboring bins. Bins in optimized-cost paths may be classified into: 1) *root bin*, 2) *intermediate bin*, and 3) *leaf bin*. *Root* is the bin which is overfilled. *Intermediate bins* are a sequence of neighbor bins between the root and leaf bins. Intermediate bins can have overfilled (supply) or available (demand) areas. The available (demand) area of intermediate bins is always lower than the inflow area. Intermediate bins always have inflow and outflow areas. The *leaf bin* is a bin which has enough space (demand) to receive inflow area without creating an overfilled bin ($demand \geq inflow$).

In Algorithm 17, the outline of the proposed network flow-based cell spreading algorithm is presented. The algorithm's input is a circuit netlist, a placement solution, and a circuit floorplan. The output is a placement solution which area density utilization is optimized.

In Line 1, the grid graph is initialized. Overlap area and cells are assigned to bins. Cells may have overlap with more than one bin. Therefore, these cells and partial cell area

Algorithm 17: Network Flow-based Cell Spreading Algorithm

```
   Data: Circuit netlist, placement solution, and circuit floorplan
   Result: Placement solution with optimized area density utilization
1  InitGridGraph();
2  OFBins = computeOverfilledBins();
3  while $!OFBins.empty()\ and\ numIts \leq maxIts$ do
4      updateMaxDisplacement();
5      for $bin \in OFBins$ do
6          if $getSupply(bin) == 0$ then
7              continue;
8          end
9          inflow = computeInflow(bin);  /* Computed as introduced in (6.1) */
10         leaf = pathAugmentation(bin, inflow);  /* Call to Algorithm 18 */
11         if $isValid(leaf)$ then
12             moveCells(leaf);  /* Call to Algorithm 20 */
13         end
14     end
15     OFBins = computeOverfilledBins();
16     numIts = numIts + 1;
17 end
```

overlap are assigned to overlapping bins. The demand and supply of bins are computed as presented in Section 3.7.

In legalization, the grid graph is built as introduced in Section 3.7. Bin height is equal to the row height. Bin width is computed as introduced in (BRENNER, 2013). Neighboring bins are connected. Bins on the opposite side of macroblocks are also connected. This grid graph may not be regular because of the macroblocks. Bin boundaries are limited to boundaries of rows and boundaries of macroblocks.

In detailed placement, the grid graph is built as introduced in Section 3.7. Bin width and height are computed in the same fashion (BRENNER, 2013). The grid graph is a regular grid of bins. Bins may have overlap to macroblock and circuit area outside of row boundaries.

In Line 2, overfilled bins are obtained. Bins in the grid graph are visited in the following order: from the left to the right and the bottom to the top. Overfilled bins are inserted in the back of a list. An overfilled bin has the supply value higher than 0. A non-overfilled bin has the supply equal to zero and demand equal to or higher than zero. In (3.22) and (3.21), supply and demand equations are presented.

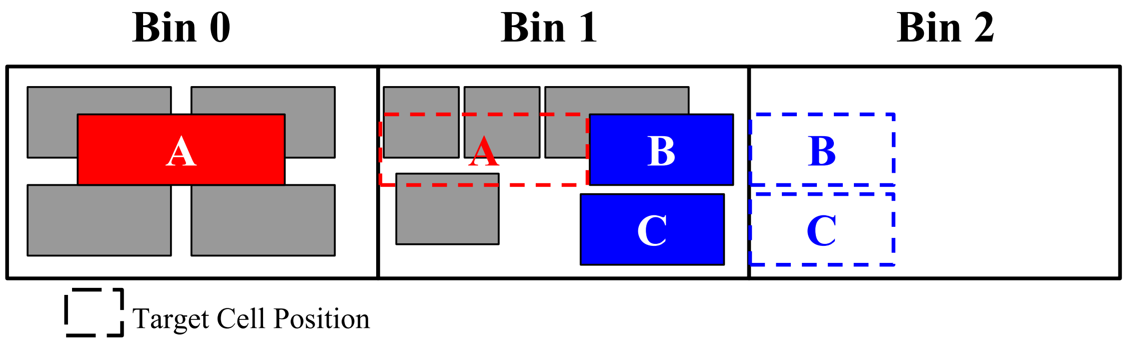
In legalization, area density constraint is defined as equal to 100% of the area utilization. Therefore, a bin is overfilled if the cell area is higher than the bin area. Bins in the non-overlapping grid graph do not have bins with the fixed area. All fixed areas are outside of these bins. In detailed placement, area density constraint is defined by the digital designer. This area constraint is the maximum area utilization that is lower than or equal to 100%. Bins of the overlapping grid graph may have part or all of their area occupied by fixed macroblocks or fixed cells. These bins may also have part or all of their area outside of row boundaries. Row boundaries on the left or right sides of the circuit placement may not be aligned.

Optimized-cost paths are searched while there are bins with area density violation. This procedure may be stopped when the maximum number of iterations is reached. The maximum number of iteration is defined to be equal to 1000. It is assumed that the cell spreading procedure will not converge (infinite lopping) if the maximum number of iterations is reached. The circuit may have regions with massive cell concentration. Moreover, high-density regions may be surrounded by macroblocks, which make the solution infeasible or extremely expensive.

In Lines 3 to 17, optimized-cost paths are searched. If valid paths have been found, cells are moved. In Line 4, maximum cell displacement constraint is computed. This cell displacement constraint avoids moving cells far away from initial positions. These cells may have considerable cell displacement. Imposing this limit minimizes adverse side effects on placement solution. In legalization, cell displacement constraint is a function of the bin width, number of iterations, α (0.6) factor and β (0.05) factor as introduced in (DARAV et al., 2017). In detailed placement, cell displacement constraints are computed for bin width and height. Detailed placement displacement constraints are computed in the same fashion of the cell displacement for bin height in legalization. The maximum cell displacement constraint is the summation of width and height cell displacement constraints.

In Lines 5 to 14, overfilled bins in the list are visited. Overfilled bins are obtained from the front of the list. The retrieved bin is checked if it has area density violation (Line 6). In Figure 6.5, an example of an intermediate bin which has fixed area density violation is presented. The optimized-cost path which is composed of bin 0, bin 1, and bin 2 and flagged cells A, B, and C fixes area density violation in bin 0 and bin 1. Bin 0 is the root bin, bin 1 is the intermediate bin, and bin 2 is the leaf bin. Flagged cells B and C provide space in bin 2, which is higher than the required space to receive cell A. In bin 2, the overfilled area is a small value. Therefore, the difference between outflow and inflow area in bin 2 is higher than or equal to the overfilled area in bin 2. In the proposed network flow-based cell spreading algorithm, usually, the outflow area is higher than the required space to be opened in the bin. The flow is the generalized type as introduced in Section 3.4.

Figure 6.5: Area overfill in bin 1 may be fixed by computing the path composed of bin 0, bin 1 and bin 2 to spread cells. Cells B and C will be moved to bin 2 to open space to receive cell A from bin 0



Source: Author (2019).

The overfilled bin is the root bin of the search tree. Therefore, an "inflow" area of the root bin must be computed. Inflow area is used in the path augmentation algorithm (Section 6.4.1) to compute outflow and to flagged cells in the root bin. The inflow area must be higher than zero and lower than or equal to the bin supply. The inflow area must also be lower than the placeable area of the neighboring bin. An inflow area, which is higher than the placeable area of the neighboring bin, will create or increase area density violation in the neighboring bin. Therefore, this inflow area automatically imposes an infeasible cost path. Computing the inflow area of the root bin is a trade-off between finding valid paths and runtime. A low inflow area in root bin increases the chances of finding valid paths, but it requires more iterations of the cell spreading algorithm, which increase runtime. The inflow area of the root bin also affects the increase of area flow in candidate paths. In intermediate bins, the outflow area is usually higher than the inflow area (generalized flow area). Therefore, while candidate paths are augmented, inflow, and outflow areas also tend to increase. On the other hand, a high inflow area decreases the chances of finding valid paths, but it reduces the number of iterations, which may decrease runtime. However, the optimized-cost path may be longer because there are bins with the capacity to receive inflow area with lower cost.

The inflow area of the root bin is computed (Line 9). The inflow area is the minimum area which must be moved out of the overfilled bin. The inflow area of the overfilled (root) bin is a way to indicate the minimum amount of area which must be moved out from the root bin. Overfilled bins can have exceeded area which varies from residual area to an immense amount of overfilled area. Inflow area of the root bin is computed as introduced in (6.1).

$$rootInflow = min\Big(\frac{totalCellArea}{numOfCells}, binSupply\Big) \tag{6.1}$$

where, $totalCellArea$ is the total area of the overlap between cells and the bin. $numOfCell$ is the total number of cells which have overlap in the bin. $binSupply$ is computed as introduced in (3.22). The inflow of the root bin is a heuristic to estimate the area flow, which is the average cell area. Therefore, at least one cell will be moved out of the root bin. Another advantage of this approach is to address multi-deck cells.

In bins with massive cell concentration, an optimized cost path is computed in each iteration of the cell spreading algorithm. In this approach, the first advantage is to increase the chances of finding optimized-cost paths. The second advantage is that optimized paths may be computed in different directions.

In Line 10, the algorithm to search the optimized-cost path is called. The path augmentation algorithm is introduced in Section 6.4.1. The path augmentation algorithm returns the leaf bin of the minimum cost path. However, path augmentation may fail to find an optimized cost path to move cells. Therefore, an invalid reference is returned.

In Line 11, it is checked if the leaf bin of the optimized-cost path is valid. In invalid-cost paths, the algorithm goes to search the optimized-cost path in the next overfilled bin. In Line 12, cells are moved between neighboring bins in the optimized cost path. The cell movement algorithm is presented in Section 6.4.3. In Line 15, the list of overfilled bins is updated. In Line 16, the number of iterations is incremented.

6.4.1 Path Augmentation Algorithm

In the path augmentation algorithm, optimized-cost paths are searched, and cells are flagged to be moved to neighboring bins. This algorithm receives an overfilled bin, circuit netlist, and grid graph. The path augmentation algorithm returns the leaf bin of the minimum cost path. In the valid paths, area density violation is minimized. On the other hand, this algorithm returns an invalid bin reference to indicate that it has failed to compute an optimized cost path.

In the search tree, each bin (node) has inflow and outflow cell areas, except leaf bins, which only have inflow cell area. In network flow, inflow and outflow definitions are introduced in Section 3.4. In the path augmentation algorithm, the area flow is provided by the set of flagged cells. For each neighboring bin, unique set of flagged cells is computed.

Restrictions in the path augmentation algorithm are presented as follows:

- Leaf bins in candidate paths must have enough free area to receive inflow area.
- Inflow cell area of a bin must be lower or equal to the placeable bin area.
- Outflow area of intermediate bins in paths must not violate inequality $outflow \geq max(inflow - demand, 0)$.

In Algorithm 18, the proposed path augmentation algorithm is presented. Branches in the search tree are pruned with branch and cut techniques. Only bins in the candidate paths which have total cell displacement lower than upper limit bound are opened. In this approach, optimized-cost paths are obtained without creating or increasing area density violation in intermediate bins. Moreover, the search space region shrinks every time a new path is found that has a lower cost than the upper limit bound.

In Line 1, the overfilled bin is inserted in the priority queue. The overfilled bin is the root bin (node) of the search tree. Each branch of the search tree is a candidate path. In the priority queue, bins are sorted by the lowest to the highest displacement cost. In the search tree, the cost of the bin is the cumulative cell displacement cost of ancestors bins. Each neighboring bin has unique set of cells to receive from the source bin. Therefore, each set of cells has a different cost.

In Line 2, the variable *bestPathBin* is initialized with null reference. Null reference of *bestPathBin* has a positive infinite cost. The reference of the leaf bin of the best path is stored in this variable. The algorithm searches an optimized-cost path while the priority queue has bins (Lines 3 to 28). The source bin is retrieved from the priority queue (Line 4). This algorithm searches a path if and only if the cost of the *source* bin is lower than the *bestPathBin* cost (Lines 5 and 6).

The *source* bin is inserted in the visited list (Line 8). Neighboring bins from the source bin are visited (Lines 9 to 18) only if these bins have not been previously visited (Lines 10 and 11). *Outflow area* is computed as shown in Section 6.4.2 and in Algorithm 19 (Line 13). Occasionally, the source bin cannot provide sufficient space to receive the inflow area. Therefore, the source bin may not provide a valid path (Line 14). An outflow area equal to zero indicates that this path is infeasible. The total cell displacement cost for a candidate path is obtained (Line 15). The tuple composed of a *neighboring* bin, cost of the candidate path and outflow area is inserted in the children list of the source bin

Algorithm 18: Path Augmentation Algorithm

```
Data: Overfilled Bin, Grid Graph, and Circuit Netlist
Result: null or valid leaf bin
1  priorityQueue.insert(overfilled bin);
2  bestPathBin = null;          /* Leaf bin of the best path */
3  while $!empty(priorityQueue)$ do
4      src = priorityQueue.top();
5      if $getCost(bestPathBin) < getCost(src)$ then
6          continue;
7      end
8      visitedBins.insert(src);
9      for $neighbor \in neighbors(src)$ do
10         if visited(neighbor) then
11             continue;
12         end
13         outflow = computeOutflow(src, neighbor);          /* Call to
            Algorithm 19 */
14         if $outflow > 0$ then
15             cost = getCost(src);
16             src.addChildrenNode(neighbor, cost, outflow);
17         end
18     end
19     for $bin \in children(src)$ do
20         if $getCost(bin) < getCost(bestPathBin)$ then
21             if $inflow(bin) \leq demand(bin)$ then
22                 bestPathBin = bin;
23             else
24                 priorityQueue.insert(bin);
25             end
26         end
27     end
28 end
```

(Line 16).

In Lines 19 to 27, the algorithm iterates in children list of the *source* bin. Bins are only visited if they have the ancestor's cell displacement cost lower than the upper limit cost (Line 20). Visited bins may have a path with a lower cell displacement cost than the current best path. The upper limit cost is the cell displacement cost of the current best path. If a child bin has available space to receive inflow area (Line 21), then the leaf bin of the best path is replaced by this child bin (Line 22). Otherwise, this child bin is inserted in the priority queue (Line 24). A bin in the priority queue may belong to a path which has the minimum cell displacement cost. Bins with cell displacement cost higher than upper limit cost are not visited. These bins belong to candidate paths that have cell displacement cost higher than the upper limit cost. Therefore, these candidate paths (branches) are pruned from the search tree.

6.4.2 Compute Outflow Area from Source Bin

In Algorithm 19, the procedure to compute the outflow area from the source bin is presented. The algorithm receives the circuit netlist, the grid of bins, the source bin, and sink (neighboring) bin. This algorithm returns the area flow to be moved out from the source bin. Otherwise, the algorithm returns zero, to indicate this algorithm has failed to find a set of cells which provide the required open space in the source bin.

Inflow area is the amount of cell area which must be available in the source bin. This area space is the minimum area necessary to receive cells without causing or increasing area density overflow. The outflow area is the amount of cell area which is going to be opened in the source bin. Outflow area of the source bin is the inflow area of the neighboring bin. Each neighboring bin has a unique outflow area and a set of flagged cells. Providing the outflow area opens enough space to receive inflow area without causing or increasing area density violation.

In legalization, the outflow area with the partial cell area is accepted to horizontal cell movements. Cells can be moved partially to a neighboring horizontal bin. In this approach, cell movement provides exact cell area space, which is required to be opened in the source bin. Horizontal cell movement in legalization is a particular case which can accurately provide the required area flow. In vertical cell movements, cells are entirely moved to the neighboring bin. In detailed placement, all moved cells out from the source bin must be placed entirely inside of the neighboring bins.

This algorithm iterates in all cells of the source bin (Lines 1 to 6). For each cell, cell displacement is computed (Line 2). Other cell restrictions may be included in this part of the code. Cell displacement cost is computed as introduced in Section 3.6.6. In the horizontal cell movement of legalization, the target position is the closest position from the current position, which gives the required area flow. Otherwise, the target position is the closest position in the neighboring bin from the source bin where the cell can be entirely placed inside of the bin boundaries. Only cells which have displacement cost lower than the maximum cell displacement restriction are inserted in the list (Lines 3 and 4). Therefore, there may not be enough cells to open the required area in the source bin.

Algorithm 19: Compute Outflow Area from Source Bin

```
   Data: Circuit netlist, grid graph, source bin, and neighboring bin
   Result: Outflow area of source bin
 1 for $cell \in listOfCells(source\ bin)$ do
 2     displacement = computeDisplacement(source, neighbor bin, cell);
 3     if $displacement < maxDisplacement$ then
 4         cellList.insert(cell);
 5     end
 6 end
 7 inflow = getInflow(sourceBin);
 8 reqFlow = max(inflow - demand(src), 0);
 9 sortCells(cellList);
10 overlapArea = 0;
11 outflow = 0;
12 for $cell \in cellList$ do
13     if $reqFlow \leq overlapArea$ then
14         return outflow;
15     end
16     overlapArea += source.getAreaOverlap(cell);
17     outflow += getArea(cell) - neighboring.getAreaOverlap(cell);
18 end
19 return 0;
```

In Line 7, the inflow of the source bin is obtained. The *source* bin can keep part of the inflow area if the source bin has available space. Required flow (*reqFlow*) is the minimum amount of cell area which must be opened in the source bin (Line 8). In this algorithm, the outflow area is usually higher than the required flow. Therefore, an additional area is commonly opened in the source bin. In some overfilled bins, this additional outflow area is enough to fix area density violation. In Figure 6.5, an example which the overfilled area is fixed by the additional outflow area of the source bin is presented.

In the list, cells are sorted from the lowest to the highest cell displacement cost (Line 9). The cell displacement cost is a negative value when the cell is going to be moved closer to the initial position. Otherwise, the cost is positive. Cells which have negative cell displacement cost have priority to be flagged. Only cells which have been moved previously may have a negative cell displacement cost.

In Lines 10 and 11, overlap area and outflow variables are initialized, respectively. All cells in the list are visited (Lines 12 to 18). Total cell area in the source bin is evaluated if it is equal to or greater than the required area flow to be opened in the source bin (Line 13). If the area mentioned above condition is valid, this algorithm returns the outflow area of the source bin (Line 14).

In the list, cells are iterated to accumulate the area overlap between these cells and the source bin (Line 16). This area overlap is the area which is going to be opened in the source bin when these cells will be moved to the neighboring bin. In horizontal cell movement of legalization, the area overlap provided by each cell is adjusted as follows: $CellAreaOverlap = min(areaOverlap(cell, sourceBin), reqFlow - overlapArea)$. The latest cell can be partially moved to the neighboring bin.

The outflow area is accumulated (Line 17). Outflow area is the area which must be available in the neighboring bin to receive these cells without causing or increasing area density violation. Outflow area from the source bin is the inflow area of the neighboring bin.

Finally, if there are not enough cells to provide white space to receive an inflow area, then this algorithm returns the zero value to indicate that it is infeasible to open the required space in the source bin (Line 19).

6.4.3 Cell Movement Algorithm

Once a path is found in Algorithm 18, selected cells must be moved between neighboring bins. The outcome of these cell movements is the reduction of area density utilization in the overfilled bin (root vertex). This area reduction in the root bin may be enough to fix the area density violation. The procedure to move cells between neighboring bins is presented in Algorithm 20. This algorithm iterates in a backtrack order. The algorithm receives the leaf bin of the valid path, grid graph, and circuit netlist. This algorithm provides a placement solution with the minimized area density violation in overfilled (root) bins.

In Line 1, the target bin variable is initialized with sink bin reference. The parent bin of the target bin is retrieved (Line 2). Outflow and inflow areas are initialized in Lines 3 and 4, respectively. Bins of the path are iterated until the root bin is reached (Lines 5 to 14). In Line 6, the inflow area of the target bin is retrieved. The previous outflow area is verified if it is lower than the computed inflow area. In neighboring bins, outflow area in the source bin must be equal to the inflow area in the target bin. Otherwise, there are shared cells which provide area flow for different pairs of source-target bins in this path.

In Figure 6.6, shared cells between neighboring bins are presented. In the path augmentation algorithm, A0, A1 and B1 bins are intermediate bins of a valid path. The

Algorithm 20: Move Cells in Minimized Cell Displacement Cost Path

```
Data: Leaf Bin, Grid Graph, and Netlist
Result: Moved cells in the path
1  target = sink;
2  source = getParent(target);
3  outflow = 0;
4  inflow = 0;
5  while $source != null$ do
6      flow = getInflowArea(target);
7      if $outflow < inflow$ then
8          flow -= (inflow - outflow);
9      end
10     outflow = moveCells(source, target, flow);
11     inflow = getInflowArea(target);
12     target = source;
13     source = getParent(source);
14 end
```

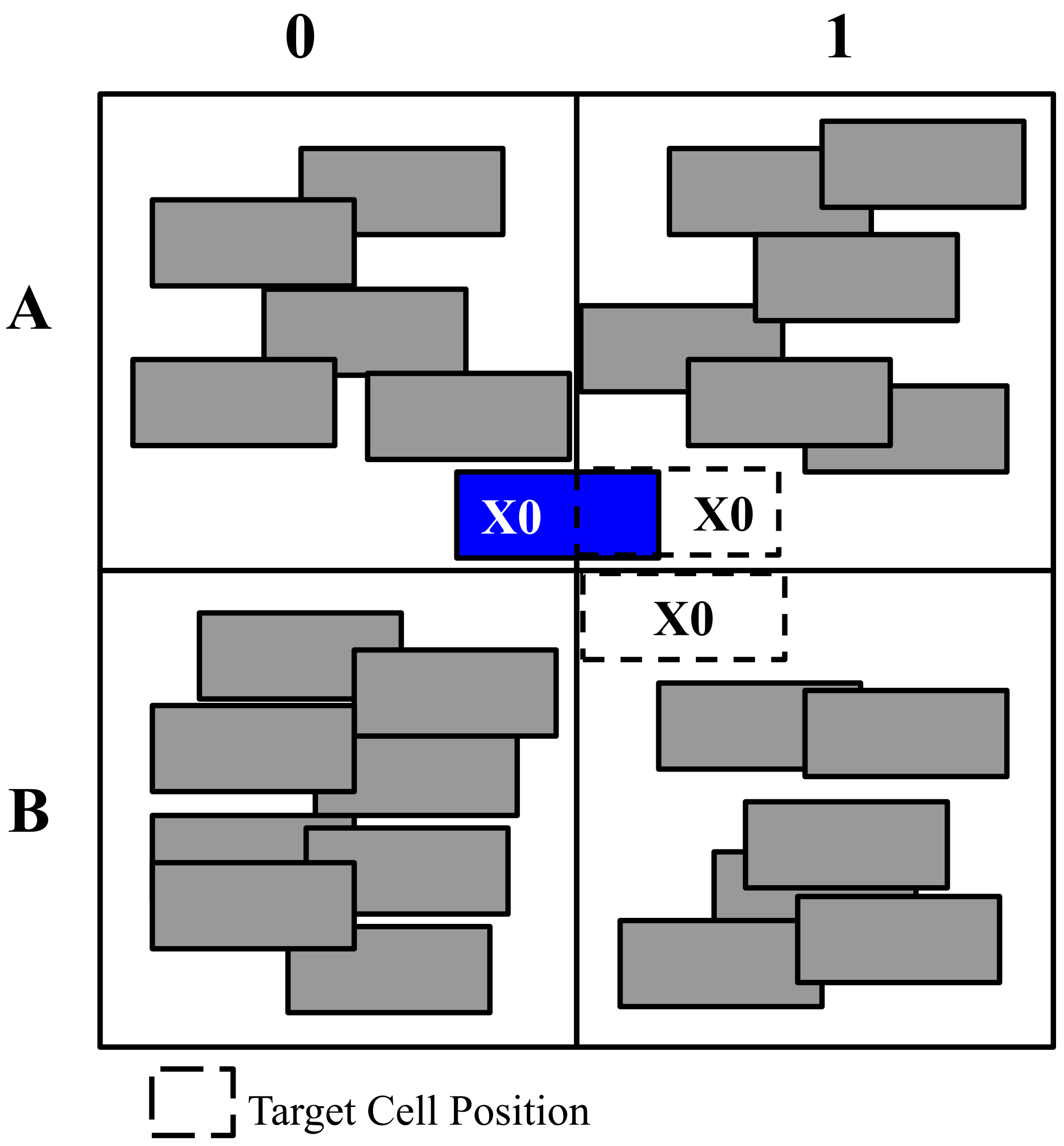
same cell *X0* provides area flow for A0-A1 and A1-A2 pairs of bins. In the cell movement algorithm, cell *X0* is moved out from bin A1 to bin B1 to provide part of the area flow. In the next iteration, the cell movement algorithm tries to move cell *X0* out from bin A0 to bin A1. However, cell *X0* is no longer available. The cell movement procedure operates in the backtrack order in the optimized-cost path.

The current area flow is adjusted (Line 8) to avoid creating or increasing area overfill in intermediate bins. Selected cells are moved to the target bin to provide outflow area (Line 10). The inflow area of the target bin is updated (Line 11). The sink bin is updated. The current source bin is the target bin in the next algorithm iteration (Line 12). The parent bin of the current source bin is updated (Line 13).

6.5 Summary

In this chapter, the proposed network flow-based cell spreading algorithm has been presented. The objective is to move cells out from high-density regions with reduced cell displacement cost. Optimized-cost paths are computed to move sets of cells between neighboring bins. The proposed cell spreading algorithm searches optimized-cost paths in n-ary trees. A search tree is built for each overfilled bin in each algorithm iteration. The root bin of the tree is an overfilled bin. Neighboring bins are inserted in a priority queue. In the priority queue, the bin with the lowest cell displacement cost is opened first. Each

Figure 6.6: The same cell is shared in two pairs of neighboring bins. The shared cell will cause mismatch in the inflow and outflow in the pairs of neighboring bins



Source: Author (2019).

branch of the tree is a candidate path. Tree branches which have cell displacement cost higher than the upper limit bound are pruned. The upper limit cost is the cell displacement cost of the minimum cost path. Every time a new path with a lower cost is found, then the minimum cost path is replaced. Therefore, the upper limit cost is reduced.

The proposed algorithm can be used to optimize area density utilization in legalization and detailed placement. In Chapter 7, the proposed cell spreading algorithm in legalization is shown. In Chapter 8, the proposed cell spreading algorithm in detailed placement is given. Experimental results for the proposed legalization and detailed placement cell spreading algorithms are discussed in Chapters 7 and 8, respectively.

7 THE PROPOSED NETWORK FLOW-BASED CELL SPREADING APPLIED IN LEGALIZATION

The proposed network flow-based cell spreading algorithm is presented in Chapter 6. In this chapter, the application of the proposed cell spreading algorithm in legalization is presented.

7.1 Introduction

Legalization is an essential and fundamental stage of the placement flow. During the legalization stage, cells are placed in positions that are aligned to boundaries of sites and rows. In these positions, cells are also free of cell overlapping. In legalization, the challenge is to move cells out from high-density regions with minimized cell displacement cost. Usually, global placement solutions have regions with high-density cell concentration. In high-density regions, legal positions may be found far away from their initial positions. Therefore, cell displacement cost can be high, which leads to significant disruption in the global placement solution.

In this chapter, the proposed Network Flow-based Legalization (NFL) algorithm is presented. The proposed NFL is a cell spreading and legalization flow. In the proposed NFL algorithm, network flow and branch and cut formal methods are integrated into the spreading cell algorithm to optimize area density utilization in legalization flow. A cost model with the history and direction of cell movements is used in the cell spreading and legalization stages. In this approach, cell displacement cost can be further optimized.

In the proposed NFL algorithm, the main contributions are summarized as follows:

- Searching optimized-cost paths with network flow and branch and cut formal methods.
- Computing cell displacement cost with a cost model, which history and direction of cell movement are easily obtained.
- Determining the direction of cell movement based on the signal (positive or negative) of displacement cost value.
- Searching legal positions in neighboring rows to minimize cell displacement cost.
- Minimizing cell displacement with both global and local view of placement solution.

- Optimizing both the average and the maximum cell displacements.

This chapter is organized as follows: In Section 7.2, the key differences to the correlated legalization algorithm are highlighted. In Section 7.3, the proposed NFL algorithm is presented. Experimental results and discussions are given in Section 7.4. In Section 7.5, the main conclusions are provided.

7.2 Correlated Network Flow-based Legalization Algorithms

The proposed NFL algorithm is in the same class of (DARAV et al., 2017; BRENNER, 2013), and BonnPlace Legalization (BRENNER, 2013) formal legalization methods. The proposed NFL has several key differences compared to the correlated legalization algorithms. In this section, the key differences are presented and discussed.

In BonnPlace Legalization, the optimized-cost paths to move cells out from high-density regions are computed with Dijkstra's algorithm (DIJKSTRA, 1959). The cost model of cell displacement only considers the current and target positions to compute cell displacement. In this cost model, It is not possible to obtain the history and direction of cell movements.

In (DARAV et al., 2017), optimized cost paths are computed using network flow and BFS techniques. Several cost paths can be computed starting from the same overfilled bin in one iteration of the search path algorithm. Different paths can share a set of the same cells in the intermediate bins to provide outflow area. If at least two of these paths are selected to provide area flow, the outflow of some paths must be adjusted. This outflow adjustment is required because the original cells are no longer available to be moved to the neighboring bins. These cells have been moved out to another location in previous paths. Iterative adjustment of path flow can easily lead to infeasible paths.

In BonnPlace Legalization and (DARAV et al., 2017), cell displacement cost model is a linear positive cell displacement cost model. In this model, the distance between the current and target positions are computed. The history and direction of cell displacements are hard to obtain. Therefore, in this cost model, cells can be moved far away from the initial positions in cumulative cell movements. In each cell movement, cell displacement cost between current and target positions can be small. Cumulative cell displacement cost can lead to substantial cell displacement. This issue arises because this cost model cannot detect if cells are going to be moved away or closer to the initial positions.

In the legalization stage of (DARAV et al., 2017), cells are legalized only inside current rows. Legal cell positions are not searched in neighboring rows. Therefore, some cells can have significant horizontal cell displacement, even if there are legal positions in neighboring rows with lower cell displacement cost.

7.3 The Proposed Network Flow-based Legalization Algorithm

In this section, the proposed NFL algorithm is presented. This proposed algorithm combines the proposed network flow-based cell spreading and cell legalization (PUGET et al., 2015) procedures. The proposed network flow-based cell spreading algorithm is presented in Chapter 6.

In legalization, the cell spreading procedure uses the non-overlap grid graph. the non-overlap grid graph is introduced in Section 3.7. During grid graph initialization, cells which have overlap with macroblocks are placed in the closest bins. Placed cells outside of row boundaries are moved to the closest bins inside of row boundaries. Cells are moved out from high-density regions using the proposed cell spreading algorithm that has been presented in Chapter 6. The proposed cell spreading algorithm is a generic technique that is applied in the legalization stage. Cell spreading and cell legalization stages have global and local views of the placement solution.

In Algorithm 21, the outline of the proposed NFL procedure is presented. The proposed algorithm receives the circuit netlist, an illegal placement solution, and circuit floorplan. NFL provides a legal placement solution with minimized cell displacement. In Line 1, the proposed network flow-based cell spreading algorithm is executed. The proposed network flow-based cell spreading has been introduced in Chapter 6. In Line 2, the legalization algorithm is executed. The cell legalization algorithm is presented in Section 7.3.1.

Algorithm 21: Circuit Legalization with the Proposed Network Flow-based Legalization Algorithm

```
  Data: Circuit netlist, illegal placement solution, and circuit floorplan
  Result: Legalized Placement Solution
1 cellSpreading() ;              /* Introduced in Chapter 6 */
2 cellLegalization() ;              /* Call to Algorithm 22 */
```

7.3.1 Cell Legalization

During the cell legalization stage, cells are aligned to site boundaries. Legal cell positions must be free of cell overlapping. In Algorithm 22, the cell legalization algorithm is presented. This algorithm is based on Jezz (PUGET et al., 2015) and Abacus (SPINDLER; SCHLICHTMANN; JOHANNES, 2008a) legalization techniques. The legalization algorithm receives the circuit floorplan, circuit netlist, and placement solution. The output from the legalization algorithm is a legalized circuit (i.e., cells are free of cell overlapping, and these cells are aligned to site and row boundaries). In Line 1, cells are sorted by x-axis. This procedure keeps the relative order of cells. All cells in the sorted list are iterated (Line 2). The legal-optimized position is searched in the current, upper, and lower neighboring rows (Line 3). The procedure to search for legal positions continues as long as the displacement cost is reduced. The best position is a legal position which has the lowest cell displacement cost. The best cell position is always replaced when a new position with a lower cell displacement cost is found. Finally, the cell is placed in the legal position (Line 4).

Algorithm 22: Legalize Circuit Cells

```
  Data: Circuit floorplan, circuit netlist, and placement solution
  Result: Legalized Circuit
1 list = sortCellsByX();
2 for $cell \in list$ do
3     pos = computeOptimizedLegalPosition(cell);
4     placeCell(cell, pos);
5 end
```

7.4 Experimental Results

In this section, experimental results of the proposed NFL algorithm are presented. The proposed NFL algorithm is evaluated using placement solutions from different global placement algorithms and distinct benchmarks. The proposed NFL algorithm has been developed in C++11 in the Rsyn framework (FLACH et al., 2017). Rsyn has been compiled with the GNU Compiler Collection (GCC). Experimental results have been conducted on an i7-4790K CPU with 32GB of memory.

The proposed NFL algorithm was evaluated from the 2006 International Symposium on Physical Design (ISPD) contest and the 2015 ICCAD contest benchmarks. The proposed algorithm is compared with Jezz (PUGET et al., 2015), FastPlace Legalizer (FPL) (VISWANATHAN; CHU, 2005) and Eh?Legalizer (Eh?L) (DARAV et al., 2017) techniques. The 2006 ISPD benchmarks have been placed using FastPlace (VISWANATHAN et al., 2007), Eh?Placer (DARAV et al., 2016) and RePlace (CHENG et al., 2018) global placement algorithms. The 2015 ICCAD benchmarks have been placer using the Eh?Placer (DARAV et al., 2016) global placement algorithm. These circuits have a significant amount of fixed macroblocks and a wide range of area density constraints. Macroblocks and area density utilization are challenges to improve the legal placement solution.

In Section 7.4.1, characteristics from the 2006 ISPD contest and the 2015 ICCAD contest benchmarks are introduced. In Section 7.4.2, results of the global placement solution with RePlace, FastPlace and Eh?Placer in benchmarks from the 2006 ISPD contest and the 2015 ICCAD contest are presented. In Section 7.4.3, experimental results of the proposed NFL and correlated legalization algorithms are shown. Complementary results of global placement solutions and legalization are given in Appendix A.

7.4.1 Characteristics of the Circuits in the Contest Benchmarks

In this section, circuits characteristics are shown. The 2006 ISPD contest and the 2015 ICCAD contest circuits are presented.

7.4.1.1 Characteristics from the 2006 ISPD Benchmarks

The 2006 ISPD contest is composed of 24 circuits. Four circuits (Bigblue3inf, Bigblue3, Newblue1, and Newblue2) have movable macroblocks. Circuits with movable macroblocks are not used to evaluate the proposed legalization algorithm. The remaining 20 circuits have from 200 thousand to 2.1 million cells. These circuits also have from 32 to 23 thousand macroblocks. In the 2006 ISPD contest benchmarks, circuits have area density constraints from 50% and 100%.

In Table 7.1, circuit characteristics from the 2006 ISPD contest benchmarks are presented. In columns 1 and 2, names and circuit acronyms are shown, respectively. In columns 3 to 7, the total number of cells (#Cells), macroblocks (#Macros), movables cells (#Movables), nets (#Nets) and IOs (#IOs) are given, respectively. In columns 8 to

10, the total area of circuit core (DA), area of fixed cells and macroblocks (FA) and area of movable cells (CA) are presented, respectively. In columns 11 and 12, the percentage of area density utilization for fixed and movable cells are presented, respectively. Metrics of fixed and movable cell area density utilization are computed by dividing fixed and movable areas by the circuit area, respectively. In column 13, the area density constraint in percentages is given. The area density constraint was established in the 2006 ISPD contest.

On average, the 2006 ISPD contest benchmarks have 658 thousand of cells and four thousand macroblocks. Area density utilization of fixed and movable cells are 56% and 22%, on average, respectively. On average, fixed area density utilization is 2.5 times higher than the movable area density utilization. In the 2006 ISPD contest benchmarks, sixteen of twenty (80%) circuits had higher fixed area density utilization than cell area density utilization. On average, the area density constraints are equal to 79%.

Table 7.1: Chracteristics of the 2006 ISPD contest circuit bechmarks

Circuits	Acronyms	#Cells $\times 10^3$	#Macros	#Movable $\times 10^3$	#Nets $\times 10^3$	#IOs	DA (mm^2)	FA (mm^2)	CA (mm^2)	$\frac{FA}{DA}$ (%)	$\frac{CA}{DA}$ (%)	Density (%)
Adaptec1inf	AD1i	211	63	211	221	480	114	49	53	43	47	100
Adaptec2inf	AD2i	255	159	254	266	407	197	121	48	62	25	100
Adaptec3inf	AD3i	452	723	451	467	0	541	333	100	62	18	100
Adaptec4inf	AD4i	496	1,329	495	516	0	541	263	108	49	20	100
Bigblue1inf	BB1i	278	32	278	284	528	114	20	61	17	53	100
Bigblue2inf	BB2i	558	23,084	535	577	0	351	135	117	38	33	100
Bigblue4inf	BB4i	1,097	1,293	1,096	1,123	0	770	514	205	67	27	100
Adaptec1	AD1	2,177	8,170	2,169	2,230	0	1,041	391	415	38	40	100
Adaptec2	AD2	211	63	211	221	480	114	49	37	43	33	60
Adaptec3	AD3	255	159	254	266	407	197	121	34	62	17	60
Adaptec4	AD4	452	723	451	467	0	541	333	70	62	13	60
Adaptec5	AD5	496	1,329	495	516	0	541	263	75	49	14	60
Bigblue1	BB1	843	646	842	868	0	541	309	115	57	21	50
Bigblue2	BB2	278	32	278	284	528	114	20	42	17	37	60
Bigblue4	BB4	558	23,084	535	577	0	351	135	82	38	23	60
Newblue3	NB3	1,097	1,293	1,096	1,123	0	770	514	144	67	19	60
Newblue4	NB4	2,177	8,170	2,169	2,230	0	1,041	391	288	38	28	60
Newblue5	NB5	330	0	330	339	337	125	0	88	0	71	80
Newblue6	NB6	442	1,277	440	465	0	642	409	143	64	22	90
Newblue7	NB7	494	11,178	483	552	0	1,951	1,545	106	79	5	80
Avg.	-	658	4,140	654	680	158	530	296	117	56	22	79

7.4.1.2 Characteristics from the 2015 ICCAD Benchmarks

The 2015 ICCAD contest is composed of eight benchmarks. This set of benchmarks has circuits with 800 thousand to 1.9 million of movable cells. The number of macroblocks is from one hundred to 4.9 thousand. Area density constraint is defined from 80% to 90% of the area utilization.

In Table 7.2, characteristics from the 2015 ICCAD contest benchmarks are presented. In columns 1 and 2, names and circuit acronyms are shown, respectively. In columns 3 to 7, number of cells (#Cells), macroblocks (#Macros), movable cells (#Movables), nets (#Nets) and IOs (#IOs) are given, respectively. In columns 8 to 10, the total area of circuit core (DA), area of fixed cells and macroblocks (FA) and the area of movable cells (CA) are presented, respectively. In columns 11 and 12, the area density utilization of fixed macroblocks and movable cells are presented, respectively. Metrics of these area density utilizations are computed by dividing fixed and movable areas by the circuit area. In column 13, area density constraint in percent is given. The area density constraint is established in the 2015 ICCAD contest.

On average, the 2015 ICCAD contest benchmarks have 1.2 million cells and 2.3 thousand macroblocks, respectively. On average, the area density utilization of fixed and movable cells are 52% and 24%, respectively. Fixed area density utilization is 2.1 times higher than movable area density utilization, on average. In the 2015 ICCAD contest benchmarks, all circuits have fixed area density utilization higher than movable cell area density utilization. On average, area density constraint is equal to 86%.

Table 7.2: Chracteristics of the 2015 ICCAD contest circuit bechmarks

Circuits	Acronyms	#Cells $\times 10^3$	#Macros	#Movable $\times 10^3$	#Nets $\times 10^3$	#IOs	DA (mm^2)	FA (mm^2)	CA (mm^2)	$\frac{FA}{DA}$ (%)	$\frac{CA}{DA}$ (%)	Density (%)
Superblue1	SB1	1,210	3,787	1,206	1,216	6,528	27	14	6	54	24	80
Superblue3	SB3	1,213	2,074	1,211	1,225	6,528	31	17	7	55	21	87
Superblue4	SB4	796	3,471	792	803	6,528	18	9	5	50	30	90
Superblue5	SB5	1,087	1,872	1,085	1,101	6,528	41	24	6	59	14	85
Superblue7	SB7	1,932	4,910	1,927	1,934	6,528	33	14	11	43	33	90
Superblue10	SB10	1,876	1,696	1,874	1,898	6,528	49	26	10	54	21	87
Superblue16	SB16	982	101	981	1,000	6,528	17	8	6	47	33	85
Superblue18	SB18	768	653	767	772	6,528	14	5	4	37	30	85
Avg.	-	1,233	2,321	1,231	1,243	6,528	29	15	7	52	24	86

7.4.2 Global Placement Results

In this section, the global placement results are presented. The global placement solution of circuits from the 2006 ISPD contest and the 2015 ICCAD contest are shown. These circuits have been placed with the RePlace, FastPlace, and Eh?Placer algorithms.

7.4.2.1 RePlace Solution from the 2006 ISPD Benchmarks

All 2006 ISPD contest benchmarks, which only have fixed macroblocks have been placed with the RePlace global placement algorithm. These circuits are placed subject to area density constraints, as presented in Table 7.1. The area density constraints of inflated (inf) circuits have been set to 90% area utilization. RePlace fails to provide global placement solution for inflated circuits when the area density utilization is set to 100%.

In Table 7.3, ABU, Non Placeable Area (NPA) and area density constraints of the 2006 ISPD circuits are presented. ABU, NPA and area density constraints are presented in percentages. In column 1, circuit acronyms are given. In columns 2 to 6, ABU of 1%, 2%, 5%, 10%, and 20% of valid bins with the highest area density violation are shown. The ABU of overfilled (Overfilled (OF)) bins is given in column 7. In column 8, ABU penalty is presented. ABU metric is computed as introduced in Section 3.6.2. In columns 9 and 10, the average and maximum NPAs are presented, respectively. NPA is the area of movable cells that have been placed inside invalid bins. A bin is invalid if it is filled by more than 80% of the fixed area. In column 11, area density constraints used in global placement are given.

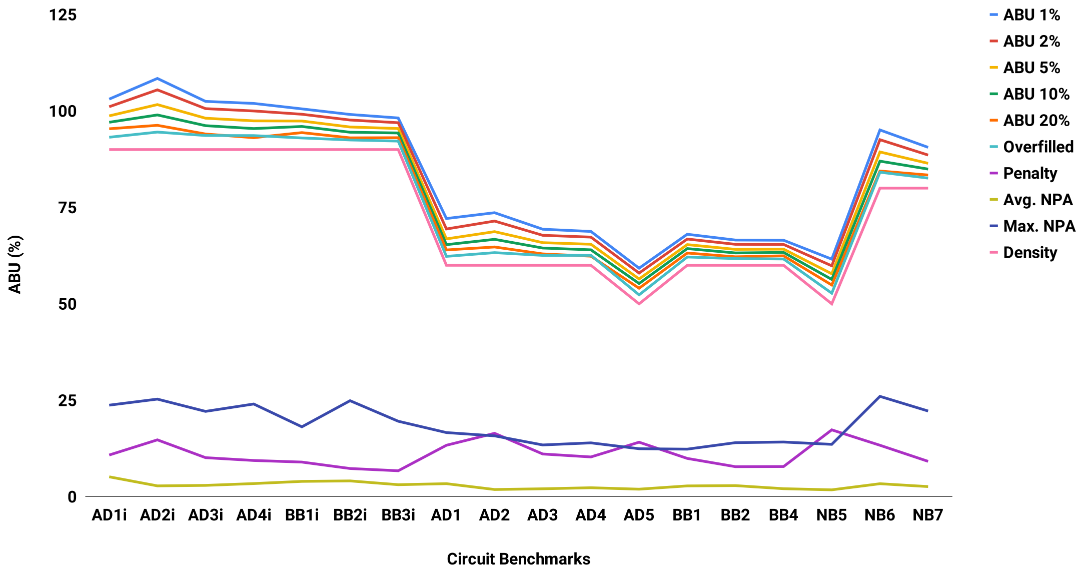
On average, in ABU 1%, RePlace provides the global placement solution with 11% higher area density utilization than the average of the area density constraint. On average, ABU of all OF bins is 3% higher than the average of the area density constraint. On average, three percent (3%) of the cell area is inside of invalid bins. Invalid bins are mainly composed of the fixed area. Therefore, NPA area indirectly indicates movable cells that have overlap with macroblocks.

In Figure 7.1, ABU distributions of the global placement solutions are presented. In abscissa, circuit acronyms are given. In ordinate, ABU in percentages is presented. In this figure, ABU 1%, ABU 2%, ABU 5%, ABU 10%, ABU 20%, ABU of overfilled bins (OF), ABU penalty (Penalty), average NPA (Avg. NPA), maximum NPA (Max. NPA), and global placement maximum area density utilization constraint (Density) are shown. ABU metrics are evaluated only for valid ABU bins.

Table 7.3: ABU, NPA and area density constraints of the 2006 ISPD circuit benchmarks are given. These circuits have been placed with the RePlace global placement algorithm. ABU, NPA and area density constraints are presented in percentages.

Circuits	ABU (%)							Avg. NPA	Max. NPA	Density (%)
	1%	2%	5%	10%	20%	OF	Penalty			
AD1i	103	101	99	97	95	93	11	5	24	90
AD2i	108	105	102	99	96	95	15	3	25	90
AD3i	102	101	98	96	94	94	10	3	22	90
AD4i	102	100	97	95	93	94	9	3	24	90
BB1i	101	99	97	96	94	93	9	4	18	90
BB2i	99	98	96	94	93	92	7	4	25	90
BB3i	98	97	95	94	93	92	7	3	20	90
AD1	72	69	67	65	64	62	13	3	17	60
AD2	74	71	69	67	65	63	16	2	16	60
AD3	69	68	66	64	63	63	11	2	13	60
AD4	69	67	65	64	62	63	10	2	14	60
AD5	59	58	56	55	54	52	14	2	12	50
BB1	68	67	65	64	63	62	10	3	12	60
BB2	67	65	64	63	62	62	8	3	14	60
BB4	66	65	64	63	62	62	8	2	14	60
NB5	62	60	58	56	55	53	17	2	14	50
NB6	95	93	89	87	84	84	13	3	26	80
NB7	91	89	86	85	83	83	9	3	22	80
Avg	84	82	80	78	77	76	11	3	18	73

Figure 7.1: ABU distributions from the 2006 ISPD benchmarks are presented. Circuits have been placed with the RePlace global placement algorithm

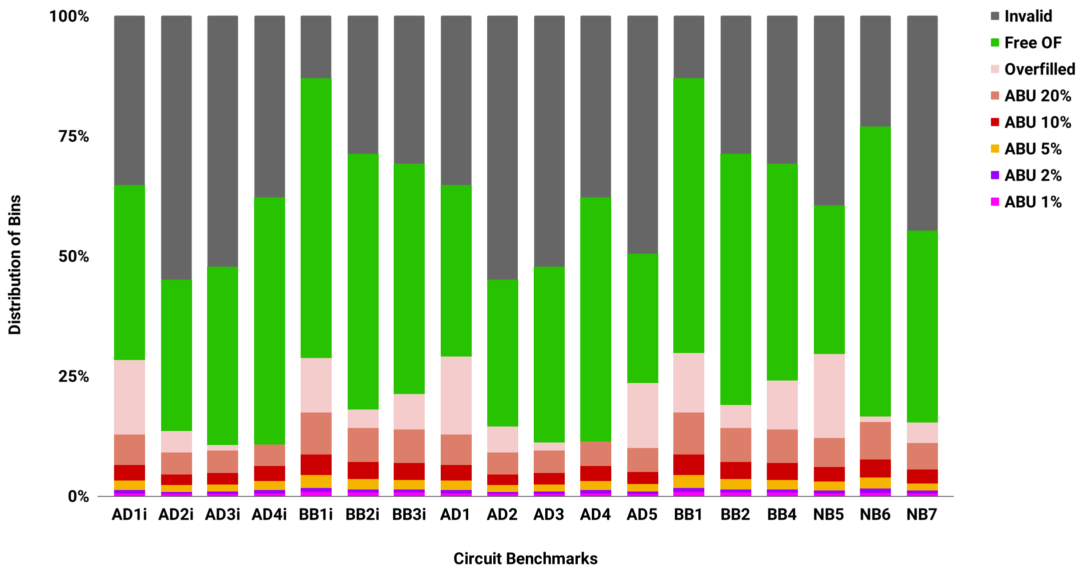


Source: Author (2019).

Circuits may be classified into two distinct groups based on ABU of overfilled bins. The first group is composed of inflated (inf) Adaptec (1 to 4), inflated Bigblue (1, 2, and 4), and Newblue (6 and 7) circuits. On average, in the first group, ABU of overfilled bins is 91%. The second group is composed of Adaptec (1 to 5), Bigblue (1, 2, and 4), and Newblue 5. On average, in the second group, ABU of overfilled bins is 60%.

In Figure 7.2, distribution of bins is presented. Bins are classified into eight categories: 1) invalid bins (Invalid), 2) non overfilled bins (Free of OF), 3) overfilled bins (OF), 4) ABU 20%, 5) ABU 10%, 6) ABU 5%, 7) ABU 2%, and 8) ABU 1%. In the abscissa, circuit acronyms are shown. In the ordinate, the percentage of the number of bins for each type are presented.

Figure 7.2: Distribution of bins with area density violation, free of area density violation and invalid bins. The 2006 ISPD contest benchmarks are placed with the RePlace global placement algorithm



Source: Author (2019).

Several circuits have a considerable amount of invalid bins (e.g., AD2i, AD3i, AD2, and AD3). This metric indicates these circuits have macroblocks which cover a significant part of the circuit core area. In Table 7.1, these circuits have 62% of the circuit area that is filled by macroblocks. Several circuits have more than 25% of the valid bins with area density violation (e.g., AD1i, BB1i, AD1, BB1, BB4, and NB5). Other circuits have less than 20% of the bins with area density violation (e.g., AD3i, AD4i, AD3, and AD4). On average, circuits have more bins which are free of overfill area than overfilled bins.

7.4.2.2 FastPlace Solution from the 2006 ISPD Benchmarks

In this section, global placement characteristics of the 2006 ISPD contest benchmarks are presented. These circuits have been placed with the FastPlace global placement algorithm. Circuits with movable macroblocks have been removed from the set of circuits used to evaluate the proposed NFL algorithm. All circuits have been placed with the maximum area density utilization established in the 2006 ISPD contest.

In Table 7.4, ABU results of the global placement solution are presented. In column 1, circuit acronyms are presented. In columns 2 to 6, ABU 1%, ABU 2%, ABU 5%, ABU 10%, and ABU 20% are shown, respectively. ABU is computed for bins with the highest area density utilization. ABU of overfilled bins (OF) and the ABU penalty are given in columns 7 and 8, respectively. ABU metric is computed as introduced in Section 3.6.2. In columns 9 and 10, the average and maximum NPAs are presented, respectively. The NPA is the average of movable cell area that is inside invalid ABU bins. An ABU bin is invalid if its area is covered by more than 80% of the fixed area. In column 11, area density constraint that was established in the 2006 ISPD contest is given.

On average, in bins of ABU 1%, FastPlace provides global placement solutions which have 82% higher cell concentration than the average area density constraint. On average, OF bins have 21% higher area density utilization compared to the average of the constraint of area density utilization. ABU of average and maximum NPAs are 5% and 45% of bin area, on average, respectively. On average, inflated bins have 45% higher area density utilization than the average maximum area density constraint. On average, non-inflated bins have 10% higher ABU than the average of the maximum area density utilization. FastPlace provides the global placement solution with significant cell concentration.

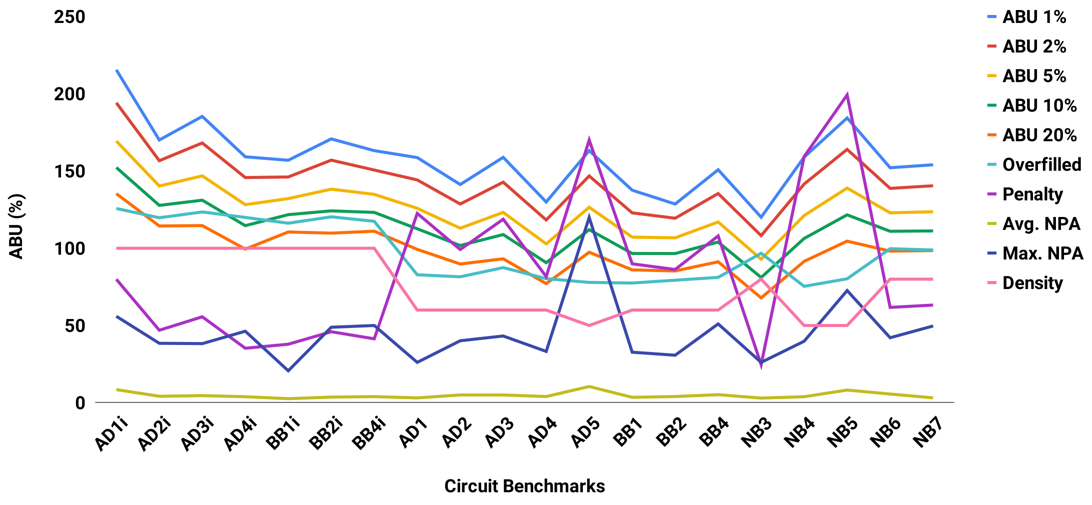
In Figure 7.2, distribution of ABU bins is presented. In abscissa, circuit acronyms are given. In ordinate, ABU in percentages is presented. In this figure, ABU 1%, ABU 2%, ABU 5%, ABU 10%, ABU 20%, ABU of overfilled bins (OF), ABU penalty (Penalty), average NPA (Avg. NPA), maximum NPA (Max. NPA), and global placement maximum area density utilization constraint (Density) are shown. ABU metrics are evaluated only for valid ABU bins.

Global placement solutions with the FastPlace algorithm have significant high-cell concentration from the global placement solutions from FastPlace. Area density constraint is not effective to establish the upper bound area density utilization. ABU penalty and maximum NPA have a high value in several circuits. These metrics indicate that sev-

Table 7.4: ABU, NPA and area density constraints of the 2006 ISPD circuit benchmarks are presented. These circuits have been placed with the FastPlace global placement algorithm. ABU, NPA and area density constraints are presented in percentages.

Circuits	ABU (%)							Avg. NPA	Max. NPA	Density (%)
	1%	2%	5%	10%	20%	OF	Penalty			
AD1i	215	194	169	152	135	126	80	8	56	100
AD2i	170	157	140	128	114	120	47	4	38	100
AD3i	185	168	147	131	115	124	56	5	38	100
AD4i	159	146	128	115	99	120	35	4	46	100
BB1i	157	146	132	122	111	116	38	3	21	100
BB2i	171	157	138	124	110	120	46	4	49	100
BB4i	163	151	135	123	111	117	41	4	50	100
AD1	159	144	126	113	99	83	123	3	26	60
AD2	141	129	113	102	90	82	99	5	40	60
AD3	159	143	123	109	93	87	119	5	43	60
AD4	130	118	103	91	77	80	82	4	33	60
AD5	163	147	127	112	97	78	170	10	120	50
BB1	138	123	107	97	86	77	90	4	33	60
BB2	129	119	107	97	85	79	86	4	31	60
BB4	151	135	117	104	91	81	108	5	51	60
NB3	120	108	93	81	68	97	25	3	26	80
NB4	159	142	121	106	92	75	159	4	40	50
NB5	184	164	139	122	105	80	199	8	73	50
NB6	152	139	123	111	98	100	62	6	42	80
NB7	154	140	124	111	99	99	63	3	50	80
Avg.	158	143	126	112	99	97	86	5	45	76

Figure 7.3: ABU, and NPA from the 2006 ISPD benchmarks are presented. These circuits have been placed with the FastPlace global placement algorithm

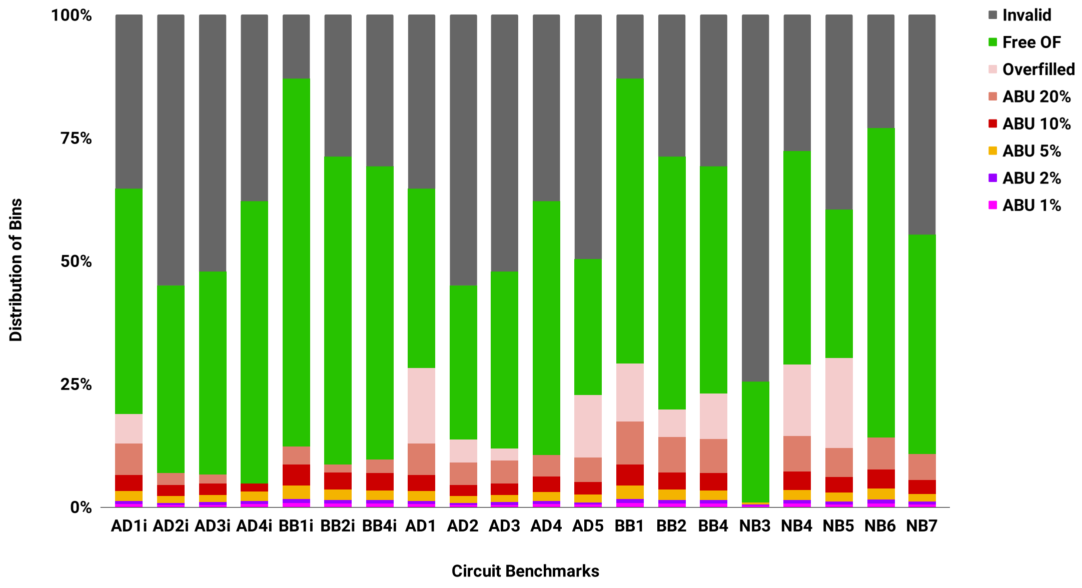


Source: Author (2019).

eral circuits have a significant cell concentration. Therefore, more cells must be moved out from high-density regions, which may increase cell displacement.

In Figure 7.4, distribution of ABU bins is presented. Bins are classified into eight categories: 1) invalid bins (Invalid), 2) non overfilled bins (Free of OF), 3) overfilled bins (Overfilled), 4) ABU 20%, 5) ABU 10%, 6) ABU 5%, 7) ABU 2%, and 8) ABU 1%. In the abscissa, circuit acronyms are shown. In the ordinate, the percentage of the number of bins are presented.

Figure 7.4: Distribution of bins with area density violation, free of area density violation and invalid ABU bins. The 2006 ISPD contest benchmarks are placed with the FastPlace global placement algorithm



Source: Author (2019).

Half of the circuits (50%) have less than 20% of the valid bins which have area density violation. Therefore, area overfill is concentrated in a small set of bins. Only four of twenty circuits have more than 25% of the total bins which have area density violation.

7.4.2.3 Eh?Placer Solution from the 2006 ISPD Benchmarks

Newblue circuits 3 to 7 from the 2006 ISPD contest benchmarks have been placed with the Eh?Placer global placement algorithm. Eh?Placer is a polar-based algorithm. Area density constraints are defined to be equal to 100%.

In Table 7.5, ABU, NPA and area density constraints are presented. ABU, NPA and area density constraints are presented in percentage. In Column 1, circuit acronyms are given. In Columns 2 to 6, ABU 1%, ABU 2%, ABU 5%, ABU 10%, and ABU 20%

of overfilled-valid bins are shown. ABU of OF bins is given in column 7. In column 8, ABU penalty is presented. The ABU metric is computed as introduced in Section 3.6.2. In columns 9 and 10, average and maximum NPAs are presented, respectively. NPA is the area of movable cells that is inside invalid bins. A bin is invalid if its area is composed more than 80% of the fixed area. In column 11, area density constraints used in global placement are given.

Table 7.5: ABU, NPA and area density constraints of the 2006 ISPD circuit benchmarks are presented. These circuits have been placed with the Eh?Placer global placement algorithm. ABU, NPA and area density constraints are presented in percentages.

Circuits	ABU (%)							Avg. NPA	Max. NPA	Density (%)
	1%	2%	5%	10%	20%	OF	Penalty			
NB3	133	122	110	101	89	113	15	6	47	100
NB4	151	137	121	111	104	112	28	9	50	100
NB5	135	126	117	110	105	108	21	10	60	100
NB6	136	126	115	108	102	109	20	8	56	100
NB7	130	122	113	108	103	107	17	8	41	100
Avg.	137	127	115	108	100	110	20	8	51	100

On average, Eh?Placer provides the global placement solution with 37% higher area density utilization in ABU 1% of bins compared to the maximum area density constraint. On average, ABU of OF bins is 10% higher than the average of area density constraints. On average, 8% of the cell area is placed in invalid bins.

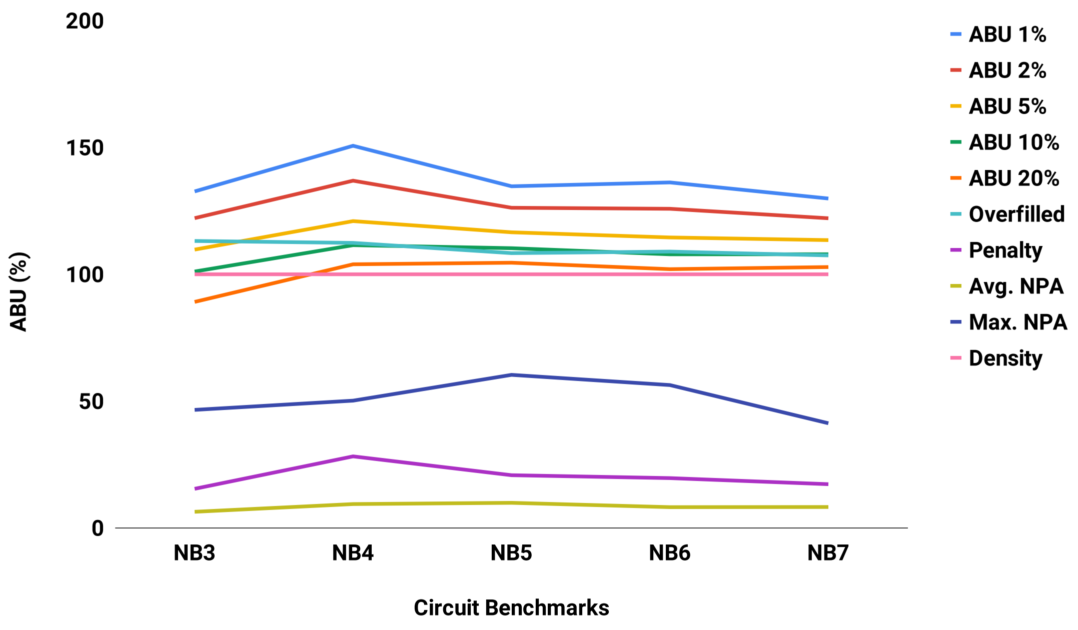
In Figure 7.5, ABU of overfilled and NPA bins are presented. In abscissa, circuit acronyms are given. In ordinate, ABU in percentage is presented. In this figure, ABU 1%, ABU 2%, ABU 5%, ABU 10%, ABU 20%, ABU of overfilled bins (Overfilled), ABU penalty (Penalty), average NPA (Avg. NPA), maximum NPA (Max. NPA), and global placement maximum area density utilization constraint (Density) are shown. ABU metrics are evaluated only for valid bins.

In Figure 7.6, the bin's distribution is presented. In this figure, invalid bins (Invalid), bins free of area density violation (Free OF), bins with area density violation (Overfilled), ABU 20%, ABU 10%, ABU 5%, ABU 2%, and ABU 1%) are given. In abscissa, circuit acronyms are shown. In ordinate, distribution of bins in percentage is presented. In these circuits, the maximum area density utilization is 100%.

7.4.2.4 Eh?Placer Solution from the 2015 ICCAD Benchmarks

The 2015 ICCAD contest benchmarks have been placed with the Eh?Placer (DARAV et al., 2016) global placement algorithm. In the global placement algorithm, the area den-

Figure 7.5: ABU from the 2006 ISPD benchmarks is presented. These circuits have been placed with the Eh?Placer global placement algorithm



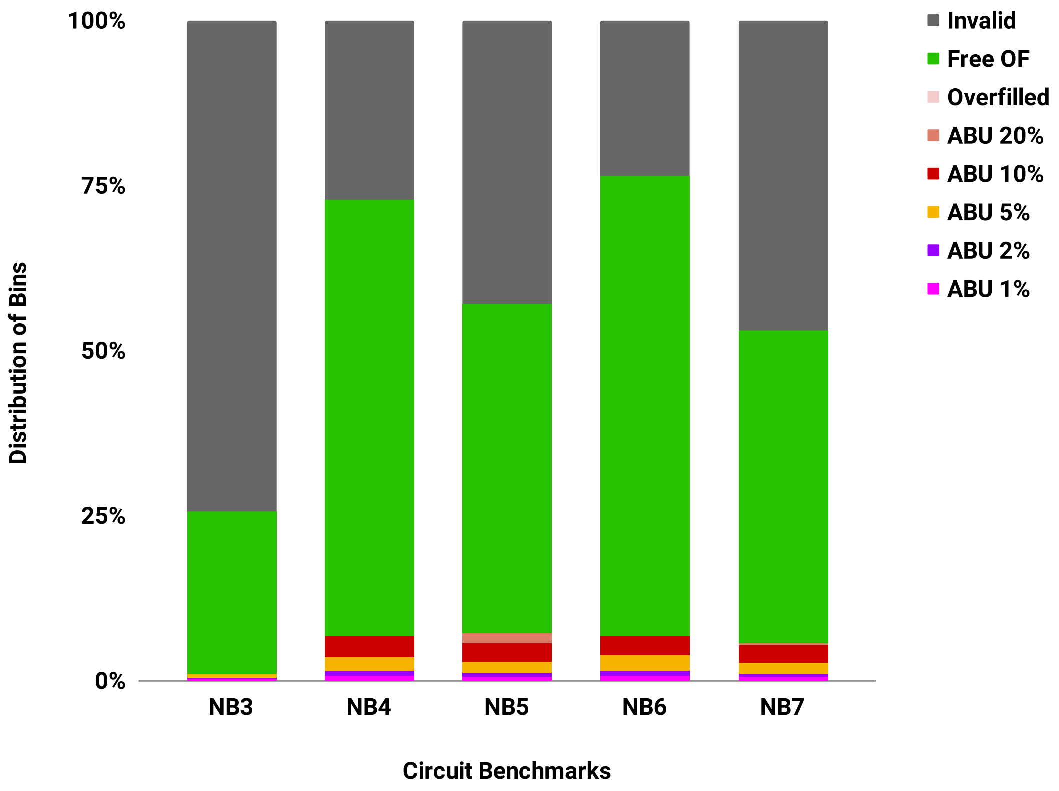
Source: Author (2019).

sity constraints are the same as the ones that are defined in the contest circuits. Except in Superblue3, area density constraint is defined as 100%. In Superblue3 circuit, the global placement algorithm fails to provide a placement solution with the area density utilization constraint lower than 100%.

In Table 7.6, ABU, NPA and area density constraints of the 2015 ICCAD circuits are presented. ABU, NPA and area density constraints are presented in percentage. In column 1, circuit acronyms are given. In columns 2 to 6, ABU of the 1%, 2%, 5%, 10%, and 20% of the bins with the highest area density violation are shown. ABU of overfilled bins is given in column 7. In column 8, ABU penalty is presented. ABU metric is computed as introduced in Section 3.6.2. In columns 9 and 10, the average and maximum NPAs are presented, respectively. NPA is the area of movable cells that is inside invalid bins. A bin is invalid if its area is composed of more than 80% of the fixed area. In column 11, the maximum area density utilization used in global placement is given.

In Figure 7.7, ABU of overfilled bins and NPA bins are presented. In abscissa, circuit acronyms are given. In ordinate, ABU in percentage is presented. Eh?Placer provides global placement solutions which are not constant in terms of area density utilization. Moreover, maximum and average NPAs indicate global placement solutions with

Figure 7.6: Distribution of bins with area density violation, free of area density violation and invalid ABU bins. The 2006 ISPD contest benchmarks have been placed with the Eh?Placer global placement algorithm



Source: Author (2019).

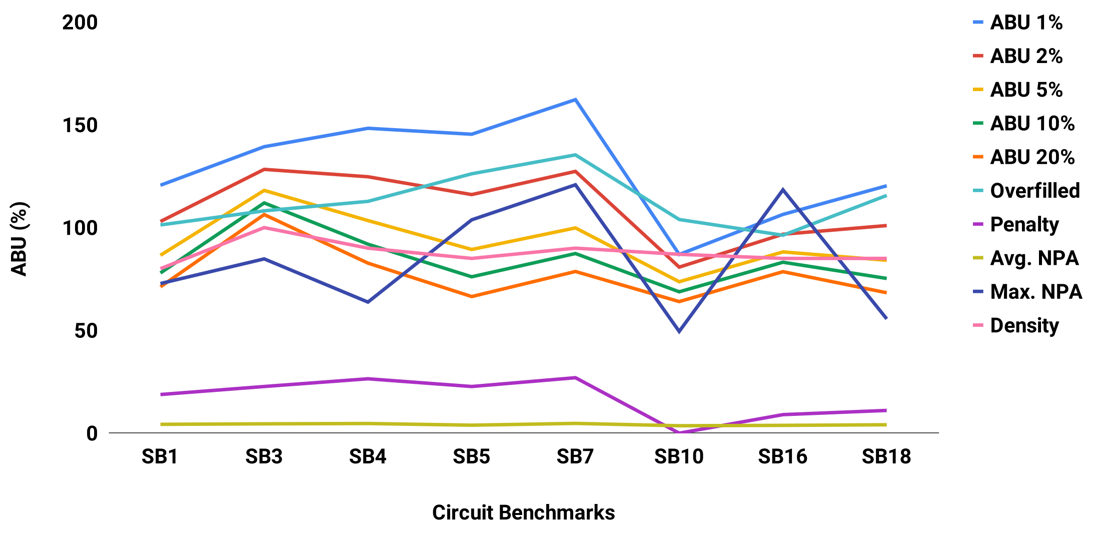
Table 7.6: ABU, NPA and area density constraints of the 2015 ICCAD circuit benchmarks are presented. These circuits have been placed with the Eh?Placer global placement algorithm. ABU, NPA and area density constraints are presented in percentages.

Circuits	ABU (%)							Avg. NPA	Max. NPA	Density (%)
	1%	2%	5%	10%	20%	OF	Penalty			
SB1	121	103	87	78	71	101	19	4	73	80
SB3	139	128	118	112	106	108	23	5	85	100
SB4	148	125	103	92	83	113	26	5	64	90
SB5	145	116	89	76	66	126	23	4	104	85
SB7	162	127	100	87	79	135	27	5	121	90
SB10	87	81	74	69	64	104	0	4	49	87
SB16	106	97	88	83	79	96	9	4	118	85
SB18	120	101	84	75	68	116	11	4	56	85
Avg.	129	110	93	84	77	112	17	4	84	88

high-cell concentration in invalid bins.

In Figure 7.8, distribution of bins in terms of area utilization from global placement solution is presented. In this figure, invalid ABU bins (Invalid), bins free of ABU violation

Figure 7.7: ABU from the 2015 ICCAD benchmarks is presented. These circuits have been placed with the Eh?Placer global placement algorithm



Source: Author (2019).

(Free OF), bins with ABU violation (Overfilled), ABU 20%, ABU 10%, ABU 5%, ABU 2%, and ABU 1% are given. In the abscissa, circuit acronyms are shown. In the ordinate, the percentage of the number of bins are presented. Essentially, circuits have a small number of bins with violation in the area density utilization.

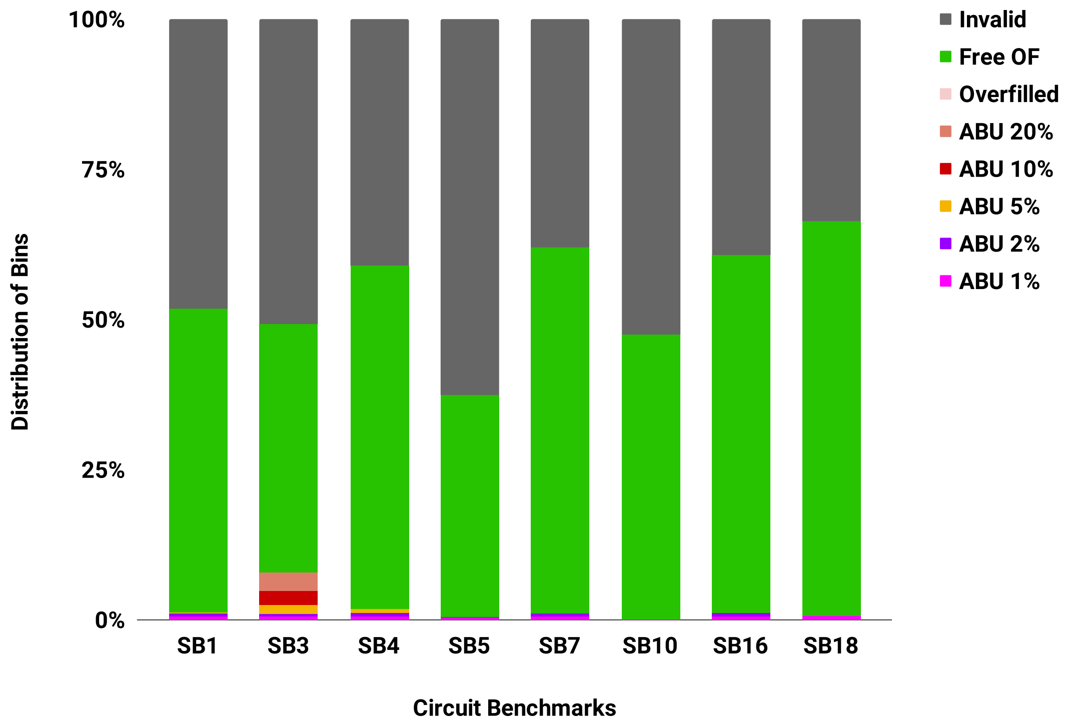
7.4.3 Experimental Results of Legalization

7.4.3.1 Legalization Results of the RePlace placement solutions from the 2006 ISPD contest benchmarks

In this Section, legalization results of the proposed NFL algorithms in the 2006 ISPD contest benchmarks are presented. Only circuits with fixed macroblocks are used to evaluate the proposed NFL algorithm. These circuits have been placed with the RePlace global placement algorithm. Legalization results of the proposed NFL algorithm are compared with FPL (VISWANATHAN et al., 2007) and Jezz (PUGET et al., 2015) legalization techniques.

In Table 7.7, legalization results are presented. In columns 1 and 2, circuit acronyms (BK) and global placement HPWL are presented, respectively. Legalization results are presented for FPL, Jezz, and the proposed NFL algorithms. In columns 3 to 5, increased HPWL in percentage from the legalization process are given. Average cell displacements

Figure 7.8: Distribution of bins with area density violation, free of area density violation and invalid ABU bins. The 2015 ICCAD contest benchmarks have been placed with the Eh?Placer global placement algorithm



Source: Author (2019).

for evaluated algorithms are presented in columns 6 to 8. Maximum cell displacements are shown in columns 9 to 11. Average and maximum cell displacements are measured in terms of row heights. The runtime of legalization techniques is given in columns 12 to 14. In columns 15 to 17, the runtime of initialization (Init.), cell spreading (CS) and cell legalization (CL) stages of the proposed NFL algorithm are presented.

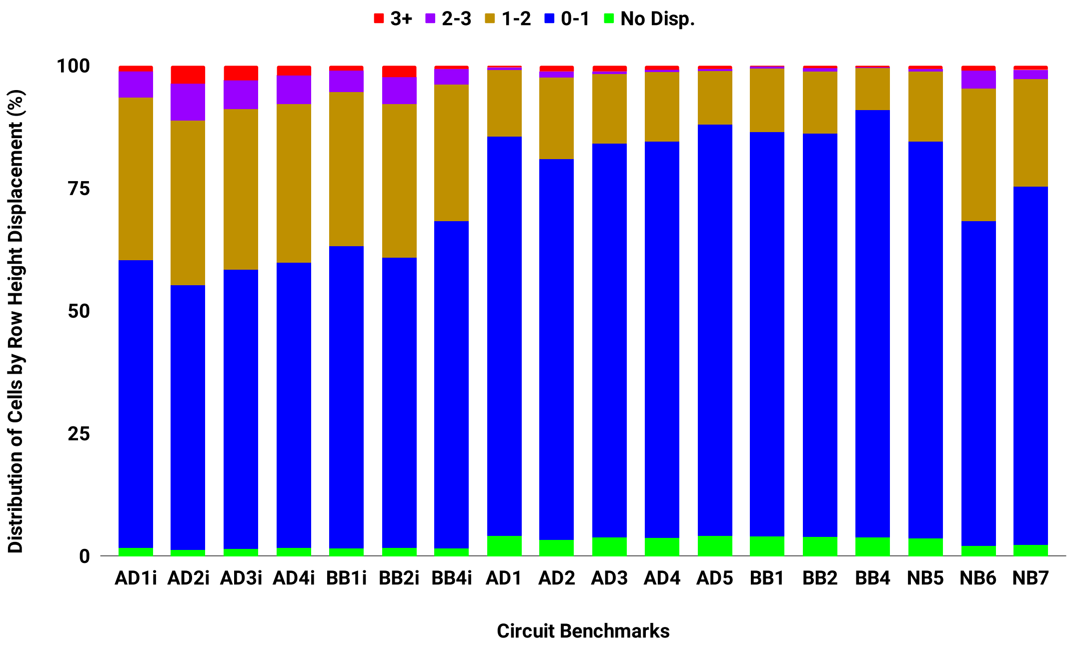
The proposed NFL algorithm achieves improvement of 10% and 3.4 times in HPWL compared to Jezz and FPL algorithms, on average, respectively. On average cell displacement, the proposed NFL obtains 10% and 5.4 times improvement compared to Jezz and FPL algorithms, respectively. In maximum cell displacement, NFL algorithm legalizes cells with 90% and 3.3 times less cell displacement than Jezz and FPL algorithms, respectively. The proposed NFL algorithm consumes 30% less runtime than FPL to legalize circuits. On the other hand, NFL requires 60% more runtime to legalize circuits compared to Jezz. On average, FPL, Jezz and the proposed NFL algorithms require $32\,\mathrm{ms}$, $10\,\mathrm{ms}$, and $24\,\mathrm{ms}$ to legalize 100 thousand cells, respectively.

Table 7.7: Experimental Results of the proposed NFL algorithm are compared with FastPlace Legalizer (FPL) and Jezz. The 2006 ISPD circuit benchmarks have been placed with RePlace algorithm.

BK	HPWL (m)	HPWL (%)			Avg. Disp. #Rows			Max. Disp. #Rows			Runtime (s)					
											FPL	Jezz	NFL			
		FPL	Jezz	NFL	FPL	Jezz	NFL	FPL	Jezz	NFL			Total	Init.	CS	CL
AD1inf	84	9.6	1.9	2.0	3.8	0.9	0.9	331	73	52	4.4	2.5	4.9	1.4	1.1	2.4
AD2inf	93	12.7	3.7	3.6	4.9	1.7	1.6	529	265	198	5.4	3.4	7.6	2.3	2.4	2.9
AD3inf	215	6.3	2.4	2.3	3.6	1.4	1.3	657	301	291	10.2	5.6	12.6	4.6	2.9	5.1
AD4inf	186	10.4	2.5	2.5	3.8	1.0	1.0	382	120	87	12.7	6.1	12.7	4.5	2.6	5.5
BB1inf	104	19.4	2.1	2.3	6.0	0.9	0.9	294	187	125	4.5	3.5	5.9	1.4	1.3	3.2
BB2inf	145	9.8	4.2	3.6	3.1	1.1	1.0	231	290	56	20.0	8.4	19.6	6.6	6.0	7.0
BB4inf	802	8.8	2.1	2.2	39.7	0.8	0.8	2305	428	127	67.9	26.9	51.8	16.6	10.5	24.7
AD1	86	2.6	0.8	0.9	1.4	0.5	0.6	88	52	52	0.4	1.0	2.5	1.4	0.1	1.0
AD2	94	3.9	2.3	1.9	2.3	1.3	1.1	252	250	197	4.2	1.3	3.7	2.3	0.2	1.2
AD3	220	2.1	1.2	1.2	1.7	0.9	0.8	298	299	299	8.4	2.3	6.8	4.4	0.3	2.1
AD4	188	2.4	1.2	1.2	1.4	0.7	0.6	257	94	87	9.0	2.5	7.1	4.5	0.2	2.4
AD5	389	1.7	1.3	1.0	1.4	0.8	0.7	297	298	291	17.6	4.0	11.0	7.0	0.2	3.7
BB1	106	1.9	0.8	0.9	1.3	0.5	0.5	93	82	36	4.3	1.2	2.6	1.4	0.1	1.2
BB2	147	3.0	2.2	1.5	1.4	0.7	0.6	153	253	51	13.7	4.2	10.1	6.6	0.3	3.2
BB4	809	1.7	1.2	1.0	1.2	0.6	0.5	401	432	127	57.6	10.9	27.8	17.1	0.6	10.1
NB5	483	2.2	1.6	1.3	1.4	0.8	0.7	552	445	128	30.4	6.0	16.8	10.6	0.5	5.7
NB6	456	7.7	1.8	1.8	2.9	0.8	0.8	561	104	55	31.4	10.4	25.9	10.7	5.4	9.8
NB7	940	5.2	2.4	1.9	2.4	1.0	0.9	636	938	287	80.8	23.0	56.6	25.7	10.0	20.8
Norm.	-	3.4	1.1	1.0	5.4	1.1	1.0	3.3	1.9	1.0	1.3	0.4	1.0	0.5	0.2	0.4

In Figure 7.9, cell displacement of the proposed NFL algorithm from the 2006 ISPD contest benchmarks is presented. In the abscissa, circuit acronyms are given. In the ordinate, the percentage of cells for each cell displacement range is shown. Cells are classified into five cell displacement groups: 1) zero cell displacement (No Disp.), 2) cell displacement higher than zero and lower than one row height (0-1), 3) cell displacement between one and two row heights (1-2), 4) cell displacement higher than two and lower than three-row heights (2-3) and 5) cell displacement higher than 3 row heights (3+). Circuits with high cell density utilization (AD1i, AD2i, AD3i, AD4i, BB1i, BB2i, and BB3i) have more legalized cells with displacement higher than 2-row heights. On the other hand, circuits with lower maximum cell density utilization have more cells legalized up to cell displacement of one-row height. Area density constraint has an impact on cell displacement in the proposed NFL algorithm.

Figure 7.9: Cell displacement from the 2006 ISPD contest benchmarks is presented. These circuits have been placed with the RePlace global placement algorithm



Source: Author (2019).

7.4.3.2 Legalization Results of the FastPlace placement solutions from the 2006 ISPD contest benchmarks

In this Section, legalization results of the proposed NFL algorithms from the 2006 ISPD contest circuits are presented. Global placement solutions are provided by FastPlace (VISWANATHAN et al., 2007) algorithm. Legalization results of the proposed NFL algorithm are compared with FPL (VISWANATHAN et al., 2007) and Jezz (PUGET et al., 2015) legalization techniques.

In Table 7.8, experimental results of the legalization stage are presented. In columns 1 and 2, circuit acronyms (BK) and HPWL of the global placement solution are presented, respectively. In columns 3 to 5, increased HPWL in percentage of FPL, Jezz and the proposed NFL algorithms are shown, respectively. Average cell displacements of legalization algorithms are given in columns 6 to 8. Maximum cell displacements after legalization techniques are shown in columns 9 to 11. Average and maximum cell displacements are measured in terms of row heights. In columns 12 to 14, runtime to legalize circuits with FPL, Jezz and NFL techniques are presented, respectively. In column 15, runtime to spread cells (CS) using the network flow-based technique in the proposed NFL algorithm is given.

The proposed algorithm achieves similar improvement on HPWL and average cell displacement compared to Jezz, on average. In HPWL and average cell displacement, the proposed NFL algorithm achieves 30% and 2.9 times improvement compared to FPL, on average, respectively. In maximum cell displacement, the proposed NFL algorithm obtains 54% and 19% improvement compared to FPL and Jezz, on average. On runtime, the proposed NFL is 10 and 25 seconds slower than FPL and jezz, on average. On average, FPL, Jezz and the proposed NFL require $42\,\mathrm{ms}$, $20\,\mathrm{ms}$ and $57\,\mathrm{ms}$ to legalize 100 thousand cells, respectively. The huge runtime in NFL is mainly caused by the legalization process of Adaptec 5 and Newblue 5. These circuits have bins with a huge NPA and ABU penalty. Therefore, cells that have overlap with macroblocks are moved to the closest valid bins of the NFL graph. Circuits with high maximum NPA and ABU penalty require significantly more runtime for all legalization algorithms. The proposed NFL algorithm is more sensitive to circuits that have high-cell concentration. The cell spreading stage will require more runtime and effort to move cells out from regions with area density violation. Computing optimized cost paths to move cells will be much more challenging in regions with high-cell concentration.

Table 7.8: Experimental Results of the proposed NFL algorithm are compared with FastPlace Legalizer (FPL) and Jezz. The 2006 ISPD benchmarks are placed with the FastPlace algorithm.

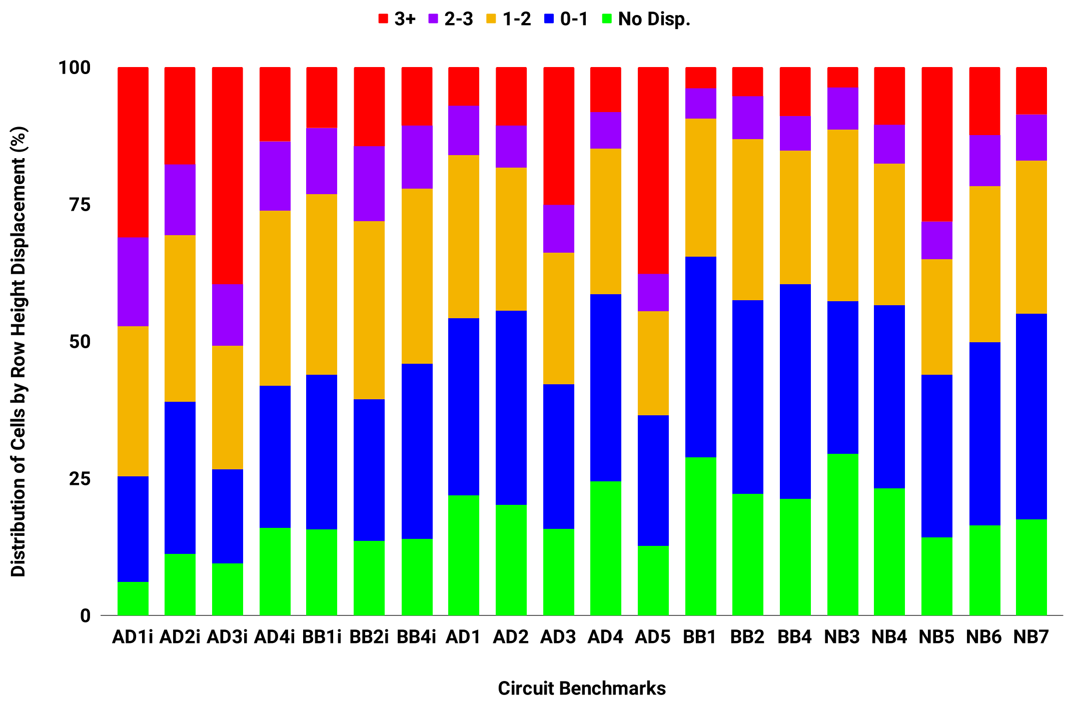
BK	HPWL (m)	HPWL (%)			Avg. Disp. #Rows			Max. Disp. #Rows			Runtime (s)			
											FPL	Jezz	NFL	
		FPL	Jezz	NFL	FPL	Jezz	NFL	FPL	Jezz	NFL			Total	CS
AD1i	92	12.2	14.6	11.6	6.9	3.6	3.5	285	431	150	6.9	3.0	8.7	6.7
AD2i	108	15.7	9.0	8.2	8.2	3.5	3.3	452	367	394	6.0	2.2	5.7	3.6
AD3i	241	16.2	28.1	41.1	11.0	10.1	13.6	905	1083	992	21.5	20.3	46.6	42.2
AD4i	225	17.4	4.9	4.5	7.1	1.7	1.7	408	210	239	11.5	3.1	6.8	3.9
BB1i	118	41.3	5.3	4.5	13.0	1.6	1.6	370	213	159	4.4	2.0	3.7	1.9
BB2i	179	8.2	6.9	6.0	4.4	1.8	1.7	490	261	81	19.0	3.9	7.5	4.2
BB4i	940	14.0	4.3	3.9	112.0	1.4	1.5	1628	231	139	68.3	15.4	32.9	18.3
AD1	93	10.8	3.1	2.9	4.3	1.1	1.1	153	70	53	3.9	0.5	1.1	0.5
AD2	101	13.8	5.4	6.3	8.3	5.1	3.8	295	317	274	4.7	0.8	3.3	2.3
AD3	224	16.9	19.0	19.8	12.2	9.4	9.0	915	822	748	17.6	12.5	24.6	21.4
AD4	214	10.2	4.4	5.0	4.9	1.8	2.1	446	300	261	12.4	1.9	4.3	2.6
AD5	377	42.1	83.9	68.8	29.1	29.2	26.5	895	1086	953	40.9	112.0	295.5	287.2
BB1	112	13.9	2.4	2.5	4.7	0.9	1.0	237	133	80	4.6	0.6	1.2	0.5
BB2	170	7.6	4.1	3.3	3.3	1.2	1.1	147	232	84	14.3	2.6	3.7	1.5
BB4	898	13.2	5.5	5.4	5.3	1.7	2.0	638	374	439	73.4	11.5	38.2	28.8
NB3	302	4.8	1.5	1.5	3.5	0.8	0.9	198	51	30	8.0	1.7	2.4	0.8
NB4	288	8.7	5.0	6.3	4.6	1.8	2.2	431	258	203	17.8	3.3	8.6	6.3
NB5	476	26.5	34.0	39.5	11.0	8.9	10.9	837	751	839	74.2	36.1	185.0	176.4
NB6	530	19.0	8.5	7.9	7.1	2.2	2.4	793	515	543	41.9	10.4	38.1	28.4
NB7	1113	9.2	4.4	4.9	4.4	1.7	1.8	633	931	571	97.4	15.6	38.0	24.6
Norm.	-	1.3	1.0	1.0	2.9	1.0	1.0	1.5	1.2	1.0	0.7	0.3	1.0	0.9

In Figure 7.10, cell displacement of the proposed NFL algorithm from the 2006 ISPD contest benchmarks is presented. In the abscissa, circuit acronyms are given. In the ordinate, the percentage of cells for each cell displacement range are shown. Cells are classified into five cell displacement groups: 1) zero cell displacement (No Disp.), 2) cell displacement higher than zero and lower than one row height (0-1), 3) cell displacement between one and two row heights (1-2), 4) cell displacement higher than two and lower than three-row heights (2-3) and 5) cell displacement higher than 3 row heights (3+).

Circuits with higher maximum cell density utilization (AD1i, AD2i, AD3i, AD4i, BB1i, BB2i, and BB3i) have more cells with cell displacement higher than 2-row heights. On the other hand, circuits with lower maximum cell density utilization have more legalized cells up to one-row height cell displacement. Maximum area density utilization has an impact on cell displacement.

The placement solution has a considerable cell concentration. The area density utilization is concentrated in a few bins. On the other hand, several bins have low area density utilization, which leads to several cells being legalized in the same positions from the global placement solution.

Figure 7.10: Cell displacement from the 2006 ISPD contest benchmarks is presented. These circuits have been placed with the FastPlace global placement algorithm



Source: Author (2019).

7.4.3.3 Legalization Results of the Eh?Placer placement solutions from the 2006 ISPD contest benchmarks

In this Section, legalization results of the proposed NFL algorithm in the 2006 ISPD contest circuits are presented. Only Newblue circuits 3 to 7 are used to evaluate the proposed NFL legalization algorithms. These circuits have the same global placement solutions used in (DARAV et al., 2017) to evaluate the Eh?Legalizer (Eh?L) algorithm. Legalization results of the proposed NFL algorithm are compared with Eh?L (DARAV et al., 2017) and Jezz (PUGET et al., 2015) legalization techniques.

In Table 7.9, experimental results of the legalization stage are presented. Eh?L results are obtained directly from the author's paper (DARAV et al., 2017). In columns 1 and 2, circuits (BK) and global placement HPWL are presented, respectively. In columns 3 to 5, increased HPWL in percentage after the legalization stage are given. Average cell displacements are shown in columns 6 to 8. Maximum cell displacements are presented in columns 9 to 11. Average and maximum cell displacements are measured in terms of row heights. In columns 12 and 13, the runtime for Jezz and NFL algorithms are presented. The runtime to initialize (init.), cell spreading (CS) and cell legalization (CL) stages of the proposed NFL algorithm are presented in columns 14 to 16, respectively. The runtime of Eh?L is not comparable with Jezz and NFL runtimes because machines are not equals.

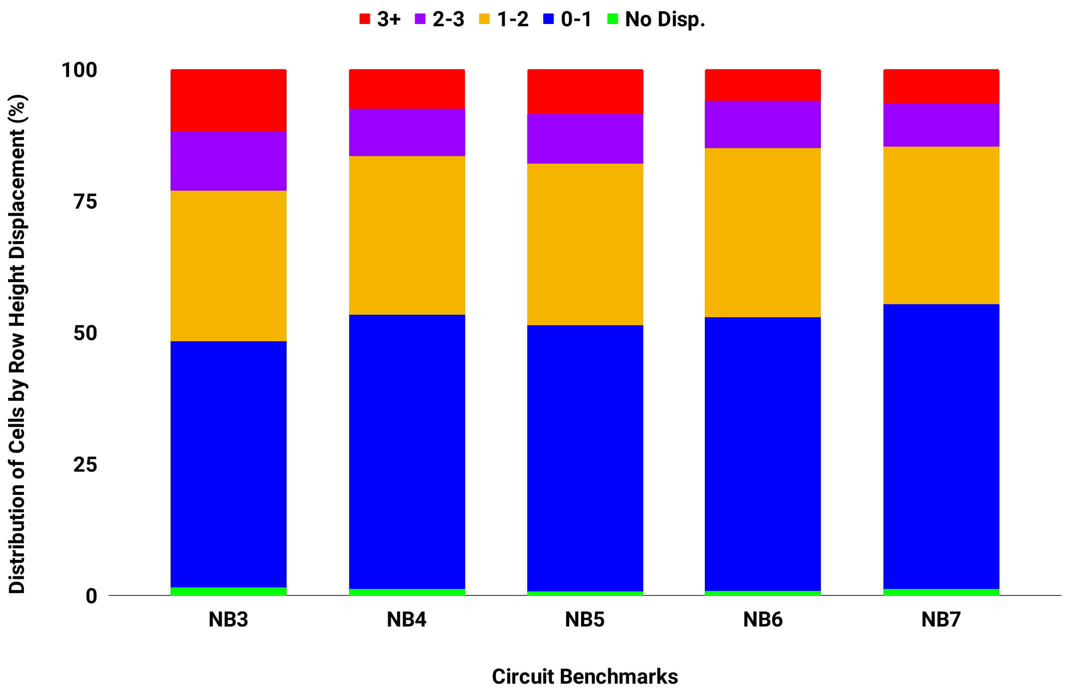
The proposed NFL algorithm obtains similar HPWL compared to Eh?L and Jezz techniques. In average cell displacement, the proposed NFL algorithm achieves 30% improvement and similar average cell displacement compared to the Eh?L and Jezz algorithms. The proposed NFL algorithm increases 10% the maximum cell displacement compared to the Eh?L technique. However, the proposed technique obtains $4.5\times$ improvement on maximum cell displacement compared to the Jezz algorithm. On average, Jezz and the proposed NFL algorithms require $8.3\,\mathrm{ms}$ and $13\,\mathrm{ms}$ to legalize 100 thousand cells, respectively.

Table 7.9: The proposed NFL algorithm is evaluated from the 2006 ISPD benchmarks. The 2006 ISPD benchmarks have been placed with the Eh?Placer algorithm. Experimental results of the proposed NFL algorithm are compared with Jezz and Eh?L algorithms.

BK	HPWL (m)	HPWL (%)			Avg. Disp. #Rows			Max. Disp. #Rows			Runtime (s)				
											Jezz	NFL			
		Jezz	Eh?L	NFL	Jezz	Eh?L	NFL	Jezz	Eh?L	NFL		Total	Init.	CS	CL
NB3	268	3.1	2.7	2.2	1.8	2.1	1.5	1153	197	197	2.5	3.3	0.6	1.1	1.7
NB4	245	2.8	2.9	2.7	1.3	1.8	1.3	219	70	83	4.1	6.5	0.6	2.7	3.2
NB5	413	3.1	3.2	3.7	1.2	1.7	1.4	426	55	92	10.4	18.1	1.2	8.9	8.1
NB6	479	2.6	2.8	2.6	1.2	1.7	1.2	161	46	46	10.6	15.2	1.1	5.9	8.2
NB7	1000	2.7	2.1	2.8	1.2	1.6	1.3	900	212	212	23.3	36.5	2.6	14.8	19.1
Norm.	-	1.0	1.0	1.0	1.0	1.3	1.0	4.5	0.9	1.0	0.6	1.0	0.1	0.4	0.5

In Figure 7.11, cell displacement of the proposed NFL algorithm from the 2006 ISPD contest benchmarks is presented. In the abscissa, circuit acronyms are given. In the ordinate, the percentage of cells for each cell displacement range is shown. Cells are classified into five cell displacement groups: 1) zero cell displacement (No Disp.), 2) cell displacement higher than zero and lower than one row height (0-1), 3) cell displacement between one and two row heights (1-2), 4) cell displacement higher than two and lower than three-row heights (2-3) and 5) cell displacement higher than 3 row heights (3+).

Figure 7.11: Cell displacement of Newblue 3 to 7 circuits from the 2006 ISPD contest benchmarks. These circuits have been placed with the Eh?Placer global placement algorithm



Source: Author (2019).

7.4.3.4 Legalization Results of the Eh?Placer placement solutions from the 2015 ICCAD contest benchmarks

In this Section, legalization results of the proposed NFL algorithms in the 2015 ICCAD contest circuits are presented. Legalization results of the proposed NFL algorithm are compared with Eh?L (DARAV et al., 2017) and Jezz (PUGET et al., 2015) legalization techniques.

In Table 7.10, legalization results from the 2015 ICCAD contest benchmarks are presented. Legalization results of the proposed NFL algorithm are compared with the Jezz algorithm. In columns 1 and 2, circuits (BK), and global placement HPWL are given, respectively. In columns 3 and 4, increased HPWL in percentage after the legalization stage are presented. Average cell displacements are shown in columns 5 and 6. Maximum cell displacements are given in columns 7 and 8. Average and maximum cell displacements are presented in terms of row heights. In columns 9 and 10, the runtime is presented for Jezz and NFL algorithms. The runtime of the cell spreading stage of the proposed NFL algorithm is shown in column 11.

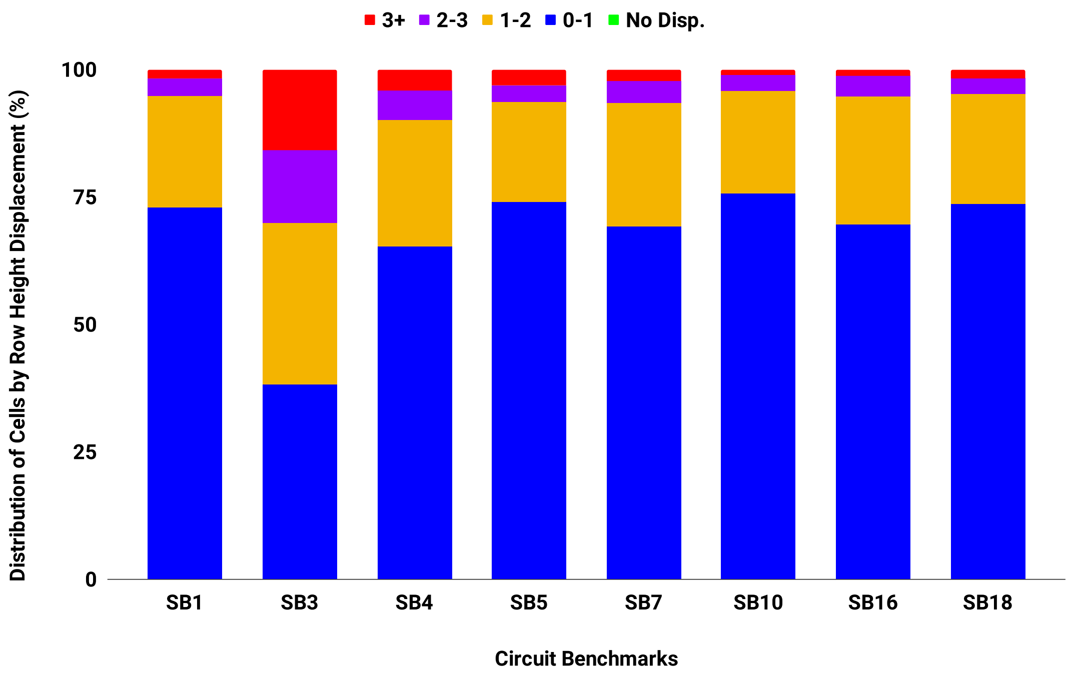
On average, the proposed NFL algorithm achieves the same HPWL and average cell displacement compared to the Jezz algorithm. In maximum cell displacement, the proposed NFL obtains two times the improvement compared to Jezz. On the other hand, NFL requires 63% more runtime to legalize circuits. On average, Jezz and the proposed NFL algorithm require $5.3\,\mathrm{ms}$ and $8.7\,\mathrm{ms}$ to legalize 100 thousand cells, respectively.

Table 7.10: Experimental Results of the proposed NFL algorithm are compared with the Jezz legalizer. The 2015 ICCAD benchmarks are placed with the Eh?Placer algorithm.

BK	HPWL (m)	HPWL (%)		Avg. Disp. #Rows		Max. Disp. #Rows		Runtime (s)		
								Jezz	NFL	
		Jezz	NFL	Jezz	NFL	Jezz	NFL		Total	CS
SB1	92	0.8	0.8	0.8	0.8	324	248	2.6	4.6	0.6
SB3	91	2.3	3.1	1.7	2.0	780	288	32.8	50.1	21.6
SB4	68	1.0	0.8	1.1	1.0	435	126	2.1	4.2	1.3
SB5	103	0.8	0.6	1.0	0.9	619	297	2.2	4.7	1.2
SB7	137	1.0	1.0	0.9	0.9	329	308	5.2	9.5	2.2
SB10	177	0.6	0.4	0.8	0.7	502	237	4.1	7.1	0.9
SB16	87	0.9	0.7	1.0	0.9	295	111	2.1	3.7	0.4
SB18	57	0.8	0.7	0.8	0.8	253	128	1.4	2.6	0.4
Norm.	-	1.0	1.0	1.0	1.0	2.0	1.0	0.6	1.0	0.3

In Figure 7.12, cell displacement is presented. In the abscissa, circuit acronyms are given. In the ordinate, cell displacements in percentages are shown. Cells are classified into five cell displacement groups: 1) zero cell displacement (No Disp.), 2) cell displacement higher than zero and lower than one row height (0-1), 3) cell displacement between one and two row heights (1-2), 4) cell displacement higher than two and lower than three-row heights (2-3) and 5) cell displacement higher than 3 row heights (3+). In all circuits, except Superblue3, nearly 75% of cells are legalized within 1-row height displacement.

Figure 7.12: Cell displacement of the proposed NFL from the 2015 ICCAD contest benchmarks is presented. Circuits have been placed with the Eh?Placer global placement algorithm



Source: Author (2019).

7.5 Summary

In this chapter, the proposed NFL algorithm is presented. The proposed cell spreading technique is applied in legalization to minimize area density violation. The objective is to legalize global placement solutions with minimized cell displacement cost. This algorithm is composed of two stages: 1) cell spreading and 2) cell legalization. In the cell spreading stage, cells are moved out from high-density regions with optimized cost paths. Optimized cost paths are computed relying on network flow and branch and cut techniques. In the cell legalization stage, cells are placed inside of boundaries of row sites. Optimized-legal positions are computed relying on a cost model which the direction and history of cell movements are easily obtained. Optimized-legal positions are also searched in neighboring rows.

The cell spreading algorithm contributes significantly to minimizing the cell displacement cost. This algorithm opens area spaces in high-density regions with minimized adverse side effects on moved cells. However, the effectiveness of this algorithm is limited by the size of bins in the grid graph. The grid graph is built with non-overlapping bins (overlapping and non-overlapping grid graphs are introduced in Section 3.7). Regions with a large cell concentration can be very challenging to optimize area density utilization. Therefore, a cell spreading algorithm with larger bins can be more useful to minimize cell concentration in very high-density utilization regions.

7.5.1 Summary of Global Placement Solutions

RePlace, Eh?Place and FastPlace algorithms provide global placement solutions. RePlace provides global placement solutions with the lowest area density violation. On average, RePlace solutions have 3% of the area density violation. On the other hand, FastPlace provides global placement solution with the highest area density violation. On average, FastPlace solutions have 21% of the area density violation. Eh?Placer provides global placement solutions for Newblue 3 to 7 circuits and the 2015 ICCAD circuits with 10% and 24% of the area density violation.

7.5.2 Summary of Legalization Solutions

The proposed NFL algorithm has been evaluated with global placement solutions. Three different global placement algorithms provide global placement solutions. The proposed NFL algorithm improves HPWL in global placement solutions that have cells evenly distributed. The proposed legalization algorithm improves also the average and maximum cell displacements compared to the state of the arts legalization algorithms. On the other hand, NFL achieves improvement in maximum cell displacement in global placement solutions which have regions with high-density utilization violations. Available space is an essential resource to the proposed NFL legalization algorithm to provide optimized-legal placement solutions. In circuits that have high-density regions, available spaces are scarce resources. The lack of white space in high-density regions and cell concentration have a significant impact on the quality of the legalization solution.

8 THE PROPOSED NETWORK FLOW-BASED CELL SPREADING APPLIED IN DETAILED PLACEMENT

In this chapter, the experimental results of the proposed spreading algorithm applied in detailed placement are presented. The proposed cell spreading algorithm is discussed in Chapter 6.

8.1 Introduction

In Very Large-Scale Integration (VLSI) physical design, placement is a challenging and critical step in the physical design flow. Placement is a crucial circuit optimization stage to achieve the required quality and performance of the circuit. Placement solution may have a considerable impact on the quality of CTS and routing solutions. Regions with high-cell concentration may cause adverse side effects in CTS and routing solutions.

The proposed NFCS has been designed using a hybrid of different algorithms to obtain the best results that are not achievable using a single method. Branch and cut and network flow formal methods have been combined. High-quality placement solutions in terms of area density utilization can be achieved in a reasonable runtime with these combined formal methods. The proposed NFCS algorithm has several advantages compared to existing cell displacement algorithms. Cells are moved out from high-density regions by computing optimized cell displacement cost paths. These cells are only moved locally between neighboring bins. Cells are flagged to be moved out of the parent's bin to the neighboring bin. These moved cells are subject to maximum cell displacement. Several cells can be moved at each algorithm iteration. The outcome of the proposed algorithm is a placement solution with reduced high-density regions. Moreover, the NFCS algorithm opens area spaces in high-density regions. These spaces may be used by optimization detailed placement algorithms to improve power consumption, timing, and routability.

The main contributions in the proposed NFCS algorithm are summarized as follows:

- A cell spreading algorithm based on network flow and branch and cut formal methods.
- A generalized flow algorithm to compute optimized cell spreading paths subject to cell displacement constraint.

- Optimizing area density utilization by moving out cells from high-density regions with minimized cost paths.
- Computing optimized paths using a cell displacement cost model which indicates the history and direction of cell movements.
- Exploring direction and the history of cell movements to further minimize cell displacement.
- Optimizing area density utilization with minimized adverse side effects on moved cells.

The work presented in this chapter has been published in *ACM Transactions on Design Automation of Electronic Systems, Vol. 24, No. 3, Article 35. May 2019* with the title **An Optimized Cost Flow Algorithm to Spread Cells in Detailed Placement**.

This chapter is organized as follows: In Section 8.2, the proposed NFCS algorithm is presented. In Section 8.3, experimental results in academic and commercial environments are presented. In Section 8.4, the chapter summary is given.

8.2 The Proposed Network Flow-based Cell Spreading Algorithm

In this section, the proposed NFCS algorithm is presented. Essentially, the NFCS algorithm is the proposed network flow-based cell spreading algorithm. The proposed network flow-based cell spreading algorithm is presented in Chapter 6.

In the proposed NFCS algorithm, the grid graph is built using bin overlap mode. Grid graph modes are introduced in Section 3.7. Width and height of bins in the grid graph are computed as introduced in Section 3.7. In the first experiment, ABU and cell spreading grid graphs are not aligned. Dimensions of bins in both grid graphs are not equal. In the second experiment, the width and height of bins in the cell spreading grid graph are equal to nine times the row height (introduced in Section 3.6.2). Therefore, both NFCS and ABU grid graphs are aligned.

In Algorithm 23, the proposed NFCS algorithm is presented. This algorithm receives circuit netlist, a placement solution with area density violation, circuit floorplan, and area density constraint. The algorithm's output is a placement solution with optimized area density utilization. Output placement solution is not legalized. In Line 1, cell spreading procedure which have been introduced in Chapter 6 is executed. The discussion of the proposed cell spreading algorithm has been presented in Chapter 6.

Algorithm 23: The Proposed Network Flow-based Cell Spreading Algorithm

```
Data: Circuit netlist, placement solution, circuit floorplan, and area density
      constaint
Result: Placement solution with optimized area density utilization
1 cellSpreading() ;                /* Introduced in Chapter 6 */
```

8.3 Experimental Results

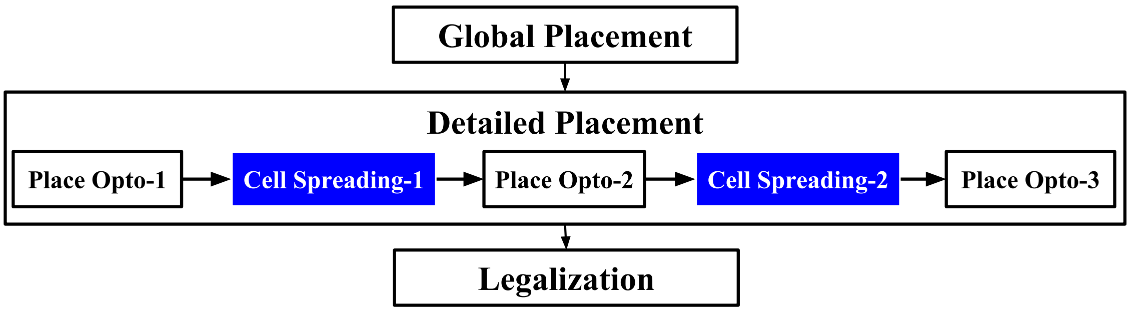
The proposed NFCS algorithm has been developed in C++. This algorithm has been evaluated in commercial and academic environments. The proposed NFCS algorithm has been analyzed with the 2006 ISPD contest and the 2014 and 2015 ICCAD contest benchmarks in the academic environment.

8.3.1 Evaluation of the proposed NFCS algorithm in Commercial Environment

The proposed NFCS algorithm has been implemented into a commercial optimization flow. Width and height of bins have been computed as introduced in Section 3.7. Bins of grid graph have overlap with fixed cells and macroblocks. The area density constraint is established to be equal to ABU 2%.

In Figure 8.1, optimization flow with the proposed NFCS algorithm in a commercial flow is presented. The proposed algorithm is inserted into two different points of the detailed placement flow. In this flow, cell movements are subject to the maximum cell displacement constraint and negative slack restrictions. The objective is to move cells out from bins which have area density utilization higher than ABU 2%. *Place Opto 1, 2, and 3* are sets of detailed placement algorithms.

Figure 8.1: The optimization flow of the proposed NFCS algorithm in commercial flow



Source: Author (2019).

Experimental results in commercial flow are presented in Table 8.1. In this flow, ABU, cell displacement, timing, power consumption, wire length (WL) and runtime metrics have been analyzed. In columns 1 to 4, circuit names (BK), the number of cells (#Cells), the number of fixed cells (#Fixed), and ABU are shown, respectively. Relative cell displacement (Disp.), WNS and TNS, leakage (LKG) and dynamic (DNC) powers, wire length (WL) and runtime are presented in columns 5 to 12, respectively.

Table 8.1: Experimental results in the commercial environment. The proposed NFCS algorithm is inserted in two stages of the commercial detailed placement flow. The two flows (with and without) the proposed NFCS algorithm are compared.

BK	#Cells $x10^3$	#Fixed	ABU	Disp. (μm)		Slack (%)		Power (%)		WL (%)	RT (%)
				Avg	Max	WNS	TNS	LKG	DNC		
c0	124	3	0.40	0.01	0.09	0.28	4.70	9.01	-0.96	-0.43	0.00
c1	58	0	2.73	0.01	3.13	1.14	19.46	-1.08	-2.90	0.16	0.00
c2	389	40	1.27	0.00	17.71	0.60	2.63	-0.77	0.36	-0.12	-7.72
c3	75	28	1.51	0.00	0.70	6.50	-2.14	0.09	0.45	0.16	0.00
c4	1,226	0	0.66	0.00	-0.27	1.16	-4.70	0.45	-1.00	0.06	1.20
c5	859	109	1.06	-0.01	1.27	-2.62	-6.58	-2.18	-0.06	-0.20	-0.76
c6	473	17	0.45	0.00	-0.14	0.16	0.37	0.64	-0.27	-0.20	1.18
c7	317	0	0.74	0.00	-2.95	-0.25	-5.16	2.71	1.27	-0.20	-5.08
c8	201	0	0.60	0.00	-1.49	0.19	7.42	0.15	0.18	-0.12	-1.67
c9	105	0	0.98	0.00	0.37	-0.73	-9.94	2.11	5.38	0.28	-1.94
c10	855	22	0.74	0.03	-0.14	0.97	25.57	0.33	0.48	-0.17	5.01
c11	1,846	1,091	0.74	-0.02	-0.39	-0.22	11.34	0.10	-0.49	0.08	3.76
c12	372	0	0.07	0.00	0.92	0.17	-1.55	-1.85	-0.31	-0.22	-3.79
c13	155	242	0.14	0.00	-0.43	-18.89	1.91	0.88	-0.41	-0.20	0.00
c14	748	0	1.00	0.00	-0.55	0.67	11.33	3.96	-0.95	-1.53	-0.71
c15	1,408	192	0.47	0.00	4.04	1.96	5.50	0.33	2.55	-0.21	-13.02
c16	99	24	2.62	0.00	0.61	1.68	29.66	1.53	-2.20	0.01	0.00
c17	36	0	0.54	0.00	0.91	-1.20	-5.99	-0.45	-0.04	0.33	0.00
c18	836	127	0.79	0.00	2.38	-0.66	-15.28	0.09	0.46	-0.11	-0.02
c19	387	72	0.90	-0.01	-0.38	-0.78	-3.09	-1.24	0.13	-0.36	-4.34
c20	278	0	0.45	0.00	0.62	-0.97	10.69	1.07	-0.14	0.05	-10.63
c21	849	0	0.81	0.01	4.76	0.96	5.72	-0.35	-0.30	-0.12	-6.16
c22	800	120	2.75	-0.01	1.37	0.44	12.08	-0.19	1.49	0.31	-1.14
c23	583	76	1.53	0.00	-0.68	0.45	20.06	0.11	0.08	0.10	0.14
c24	629	508	1.08	0.00	-0.10	-0.48	1.09	-0.07	3.28	-0.16	4.68
c25	753	113	2.64	0.00	-0.02	-0.39	-8.70	0.35	0.13	0.00	3.25
c26	644	48	1.26	0.00	-2.09	-0.30	6.23	-0.22	-0.14	0.05	-3.17
c27	573	0	0.43	0.00	-2.64	-1.29	-6.32	15.18	0.08	0.20	1.40
c28	921	15	1.05	0.00	-1.67	0.51	8.49	0.41	-0.40	-0.38	-3.12
c29	38	434	1.98	-0.03	0.27	0.91	-0.83	-0.10	0.06	0.10	0.00
c30	117	0	0.43	0.01	0.00	0.02	2.48	-2.24	-1.18	0.30	0.00
c31	42	1	1.05	0.00	0.08	0.26	-1.27	2.67	1.60	1.23	0.00
avg	525	103	1.06	0.00	0.79	-0.30	3.60	0.98	0.19	-0.04	-1.33

On average, an improvement of 1.06, in ABU, leads to the detailed placement algorithm's improvement by 3.6% of TNS and 0.98% on leakage power. TNS and WNS are improved on 19 (60%) of 32 circuits. Detailed placement algorithms may find improved positions to move cells inside of the high-density regions. The requirement for bigger and more cells and buffers to achieve circuit performance is alleviated. The impact on the runtime of the entire detailed placement flow is not significant. In several circuits, the runtime of the flow has been reduced.

8.3.2 Evaluation of the proposed NFCS algorithm in academic circuits

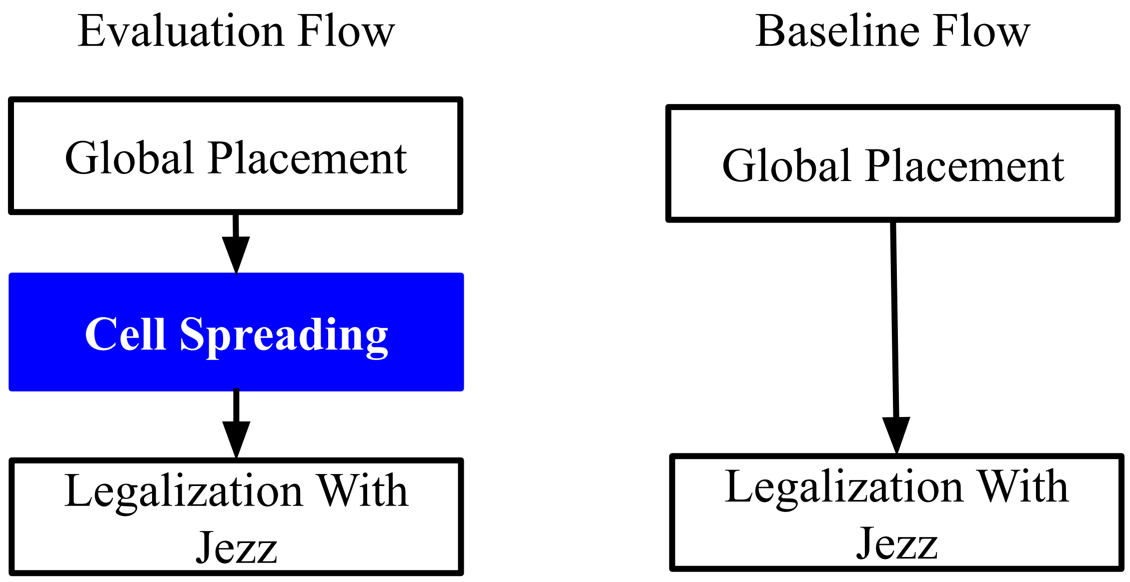
In academic environment, the proposed NFCS algorithm has been developed in C++-11 in the Rsyn framework (FLACH et al., 2017). Circuits from the 2006 ISPD contest, 2014 ICCAD contest and 2015 ICCAD contest are used to evaluate the proposed NFCS algorithm. The 2006 ISPD contest and the 2015 ICCAD contest have been placed with FastPlace (VISWANATHAN et al., 2007) and Eh?Placer (DARAV et al., 2016) (Polar-based(LIN et al., 2013)) global placement algorithms, respectively. In the 2006 ISPD contest, only circuits that have fixed macroblocks have been removed. Global placement solution of circuits from the 2006 ISPD contest are shown in Chapter 7 in Section 7.4.2.2. In these academic circuits, area density constraint is defined in the contest benchmarks. These circuits have been placed considering the area density constraint.

The proposed NFCS algorithm is evaluated in an optimization flow with the Jezz (PUGET et al., 2015) legalization algorithm. In Figure 8.2, the evaluation flow of the proposed cell spreading in the academic environment is presented. In this flow, cells are moved out from high-density regions using the proposed NFCS algorithm. Then, circuits are legalized with the Jezz algorithm. This flow is named Cell Spreading Jezz (CSJ). Legalization results of this flow are compared with legalization results only from the Jezz algorithm. Jezz is very sensitive to white space in high-density regions. Therefore, Jezz can further explorer spaces that have been opened by the proposed NFCS algorithm. The Jezz algorithm highlights that opening white spaces in high-density regions improve placement solution. In the CSJ flow, Jezz may be replaced by detailed placement algorithms to optimize wire length, timing violations, power consumption, and routability.

The proposed NFCS algorithm is also evaluated considering aligned and non-aligned cell spreading and ABU grid graphs. In the non-aligned grid graph, the width and height of bins are computed as introduced in Chapter 6. In the ABU grid, the width and height of bins are equal to nine (9) times row height. In Figure 8.3, non-aligned cell spreading and ABU grid graphs are shown. In aligned grid graphs, the width and height of bins in cell spreading and ABU grid graphs are equal to nine (9) times row height.

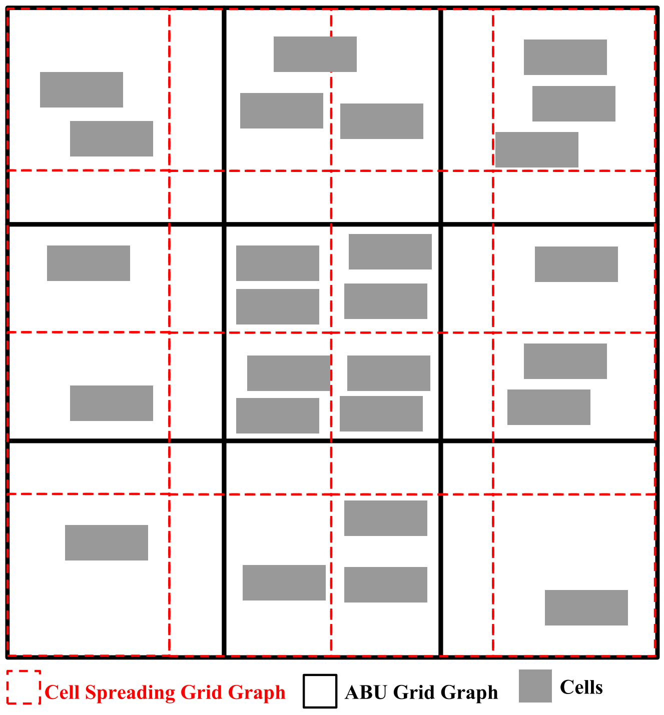
In aligned and non-aligned grid graphs, experimental results of the proposed NFCS

Figure 8.2: Experimental configuration of the proposed cell spreading algorithm in academic environment



Source: Author (2019).

Figure 8.3: Non-aligned cell spreading and ABU grid graphs



Source: Author (2019).

from the 2006 ISPD benchmarks are evaluated. Experimental results with non-aligned and aligned cell spreading and ABU grid graphs are presented in Sections 8.3.2.1 and 8.3.2.2, respectively. The area density restriction is defined to be equal to 100% due to the high-cell concentration in global placement solution. In non-aligned grid graphs, ABU improvement, Average Cell Displacement (ACD), and legalization results are presented. In aligned grid graphs, only ACD and legalization results are shown.

8.3.2.1 Experimental Results in benchmarks from the 2006 ISPD contest in non-aligned grid graphs

In Table 8.2, ABU improvement is presented. Placement improvements of ABU are obtained by executing the proposed NFCS algorithm in non-aligned ABU and cell spreading grid graphs. ABU and NPA results are evaluated after executing NFCS in benchmarks from the 2006 ISPD contest. In column 1, circuit acronyms are given. In columns 2 to 6, ABU of the valid bins with the highest area density violation are shown. ABU of the overfilled (OF) bins is presented in column 7. In column 8, ABU penalty is highlighted. ABU is computed as introduced in Section 3.6.2. In columns 9 and 10, average and maximum NPA are given. NPA is the cell area inside of invalid bins. A bin is invalid if it has more than 80% of its area covered by fixed macroblocks or fixed cells. Area density constraints are presented in column 11.

In Table 7.4 from Chapter 7, ABU of the global placement solution with FastPlace is presented. In this experiment, area density restriction is set to 100% of bins area for all circuits. ABU results achieved with the proposed NFCS algorithm are compared with ABU from global placement solutions. On average, in ABU 1%, area density violation has been reduced by 23% compared with global placement solution (Table 7.4 from Chapter 7). ABU of OF bins and ABU penalty have been minimized by 7% and 16%, on average, respectively. ABU is computed using the area density constraint established in the 2006 ISPD contest.

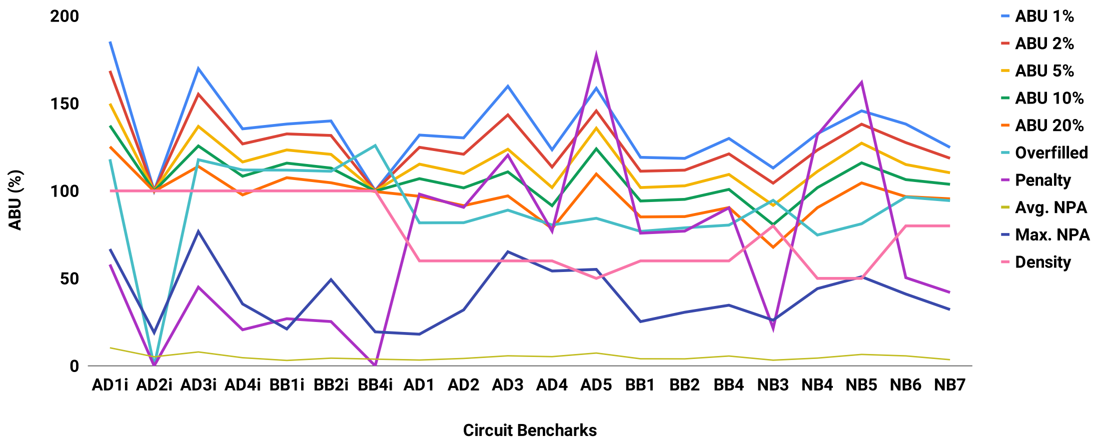
In Figure 8.4, ABU distributions of the proposed NFCS algorithm are presented. In abscissa, circuit acronyms are given. In ordinate, ABU in percentage is presented. In the figure, ABU 1%, ABU 2%, ABU 5%, ABU 10%, ABU 20%, ABU of OF bins (Overfilled), ABU penalty (Penalty), average NPA (Avg. NPA), maximum NPA (Max. NPA), and global placement maximum area density utilization constraints (Density) are shown. ABU metrics are evaluated only for valid ABU bins.

In Table 8.3, ACD of the proposed NFCS algorithm in non-aligned grid graphs is presented. Cell displacement is presented in the number of row heights. In column 1, circuit acronyms are presented. ACD of cells with the highest cell displacements are given in columns 2 to 5. In columns 6 and 7, the average and maximum cell displacements are shown, respectively. The number of moved cells and the total number of cells are given in

Table 8.2: ABU improvement from the proposed NFCS algorithm in non-aligned ABU and cell spreading grid graphs. ABU, NPA and area density restriction (density) are presented in percentage.

Circuits	ABU (%)							Avg. NPA	Max. NPA	Density (%)
	1%	2%	5%	10%	20%	OF	Penalty			
AD1i	185	169	150	137	125	118	58	10	67	100
AD2i	100	100	100	100	100	0	0	5	19	100
AD3i	170	155	137	126	114	118	45	8	77	100
AD4i	135	127	117	108	98	112	21	5	35	100
BB1i	138	133	123	116	108	112	27	3	21	100
BB2i	140	132	121	113	105	111	25	4	49	100
BB4i	100	100	100	100	100	126	0	4	19	100
AD1	132	125	115	107	97	82	98	3	18	60
AD2	130	121	110	102	92	82	91	4	32	60
AD3	160	143	124	111	97	89	120	6	65	60
AD4	123	114	102	92	78	81	77	5	54	60
AD5	159	146	136	124	110	84	177	7	55	50
BB1	119	111	102	94	85	77	76	4	25	60
BB2	119	112	103	95	85	79	77	4	31	60
BB4	130	121	109	101	90	80	90	6	35	60
NB3	113	104	92	81	68	95	22	3	26	80
NB4	133	123	111	102	90	75	132	4	44	50
NB5	146	138	127	116	105	81	162	7	51	50
NB6	138	128	115	106	97	96	50	6	41	80
NB7	125	119	110	104	95	94	42	4	32	80
Avg.	135	126	115	107	97	90	70	5	40	76

Figure 8.4: ABU distributions from the 2006 ISPD circuits are presented. The cell spreading and ABU grid graphs are not aligned. Circuits have been placed with the FastPlace global placement algorithm



Source: Author (2019).

columns 8 and 9, respectively. In column 10, the percentage of moved cells are presented.

On average, the 2% of cells with the highest cell displacement have the displace-

Table 8.3: The ACD of the proposed NFCS algorithm in non-aligned ABU and cell spreading grid graphs. The ACD is presented in number of row heights.

Circuits	ACD (Row Heights)				Avg. Disp.	Max. Disp.	Cells ($\times 10^3$)		%
	2%	5%	10%	20%			Moved	Total	
AD1i	6.2	4.0	2.6	1.5	0.3	47.7	33	211	15.8
AD2i	41.9	18.2	9.6	4.9	1.0	195.4	35	254	13.9
AD3i	146.1	83.4	48.7	26.9	5.6	443.0	150	451	33.3
AD4i	4.3	2.4	1.3	0.6	0.1	86.8	37	495	7.5
BB1i	2.9	1.7	0.9	0.4	0.1	37.0	17	278	6.0
BB2i	4.0	2.2	1.2	0.6	0.1	53.0	41	535	7.7
BB4i	3.5	2.0	1.1	0.5	0.1	124.0	158	2,169	7.3
AD1	2.7	1.3	0.6	0.3	0.1	45.7	8	211	3.7
AD2	87.3	44.4	22.7	11.4	2.3	197.0	27	254	10.6
AD3	159.4	88.5	49.3	25.9	5.2	443.6	105	451	23.3
AD4	30.3	13.6	6.9	3.4	0.7	163.7	31	495	6.3
AD5	309.1	216.5	149.0	89.1	18.6	747.0	339	842	40.3
BB1	5.0	2.0	1.0	0.5	0.1	78.3	6	278	2.1
BB2	9.0	3.7	1.9	0.9	0.2	52.8	19	535	3.6
BB4	27.5	13.6	7.1	3.6	0.7	147.4	195	2,169	9.0
NB3	0.4	0.2	0.1	0.0	0.0	26.0	4	483	0.8
NB4	24.1	11.4	5.9	3.0	0.6	113.2	51	643	7.9
NB5	156.5	98.8	60.6	33.3	6.8	447.3	349	1,228	28.4
NB6	36.0	17.7	9.5	4.8	1.0	293.8	150	1,248	12.0
NB7	26.0	11.2	5.8	2.9	0.6	241.5	205	2,481	8.3
Avg.	54.1	31.8	19.3	10.7	2.2	199.2	98	786	12.4

ment of 54 row heights. The average and maximum cell displacements are 2.2 and 199 row heights, on average, respectively. On average, 12% of all movable cells are moved to minimize area density violations. Adaptec 5 (AD5) has a huge cell displacement. This circuit has a huge NPA (Table 7.1).

In Table 8.4, Legalization results of Jezz and CSJ flows are presented. The impact of the proposed NFCS in legalization is evaluated with the Jezz algorithm. In column 1, circuit acronyms (BK) are presented. HPWL of global placement solutions are given in column 2. Increased HPWL of CSJ and Jezz flows are presented in columns 3 and 4, respectively. In columns 5 to 8, average and maximum cell displacements are shown. Runtime of Jezz and CSJ flows are highlighted in columns 9 and 10, respectively. In columns 11 and 12, the runtime of cell spreading and legalization stages of the proposed NFCS are presented, respectively.

The CSJ flow achieves 19% improvement on HPWL compared to Jezz, on average. In average and maximum cell displacement, CSJ flow minimizes cell displacement by 16% and 10% compared to Jezz, on average, respectively. Achieved improvement is

Table 8.4: Legalization results from Jezz and CSJ flows in non-aligned ABU and cell spreading grid graphs.

BK	HPWL (m)	HPWL (%)		Avg. Disp. #Rows		Max. Disp. #Rows		Runtime (s)			
								Jezz	NFL		
		Jezz	CSJ	Jezz	CSJ	Jezz	CSJ		Total	CS	Legal
AD1i	92	14.6	14.1	3.6	3.6	423	340	8.4	23.3	15.2	8.1
AD2i	108	8.9	8.6	3.5	2.9	366	557	6.4	22.4	15.1	7.3
AD3i	241	27.7	23.5	10.1	9.2	1032	991	51.0	142.0	115.9	26.0
AD4i	225	4.9	4.4	1.7	1.6	208	154	8.8	19.9	11.6	8.4
BB1i	118	5.4	4.7	1.6	1.5	209	194	5.7	13.6	8.1	5.5
BB2i	179	6.9	6.3	1.8	1.7	284	260	9.7	22.7	13.6	9.1
BB4i	940	4.3	4.0	1.4	1.4	231	206	39.9	94.3	55.6	38.6
AD1	93	3.0	3.0	1.1	1.1	74	61	1.7	3.4	1.8	1.6
AD2	101	5.4	6.6	5.1	3.5	316	480	2.6	12.2	8.8	3.5
AD3	224	19.0	14.6	9.4	7.6	823	659	32.6	69.8	57.0	12.8
AD4	214	4.4	4.1	1.8	1.7	298	274	5.4	14.7	9.9	4.8
AD5	377	83.1	60.2	29.1	22.1	1078	904	283.3	793.3	731.7	61.5
BB1	112	2.4	2.4	0.9	0.9	143	125	2.0	3.4	1.5	1.9
BB2	170	4.1	3.4	1.2	1.1	232	72	6.1	13.3	7.7	5.6
BB4	898	5.5	4.9	1.7	1.8	376	415	30.3	129.7	102.0	27.7
NB3	302	1.5	1.5	0.8	0.8	52	38	4.0	6.4	2.7	3.7
NB4	288	5.0	4.4	1.8	1.7	250	202	9.3	34.6	27.2	7.4
NB5	476	34.3	30.2	8.9	9.2	753	751	89.0	480.7	438.7	42.0
NB6	530	8.5	7.0	2.2	2.3	522	640	27.5	147.4	119.6	27.8
NB7	1113	4.5	4.6	1.7	1.7	931	518	39.0	111.4	78.2	33.2
Norm.	-	1.19	1.00	1.15	1.00	1.10	1.00	0.31	1.00	0.84	0.16

obtained by opening white space in high-density regions. Therefore, Jezz in CSJ flow smartly explores these newly opened space.

8.3.2.2 Experimental Results in benchmarks from the 2006 ISPD contest in aligned grid graphs

In Table 8.5, ABU improvement of the NFCS algorithm in aligned grid graphs is presented. Placement improvements of ABU are obtained by executing the proposed NFCS algorithm in aligned ABU and cell spreading grid graphs. ABU and NPA results are evaluated after executing NFCS in benchmarks from the 2006 ISPD contest. In column 1, circuit acronyms are given. In columns 2 to 6, ABU of valid bins with the highest area density violation are shown. ABU of the overfilled (OF) bins is presented in column 7. In column 8, ABU penalty is highlighted. ABU is computed as introduced in Section 3.6.2. In columns 9 and 10, average and maximum NPA are given. NPA is the cell area inside invalid bins. A bin is invalid if it has more than 80% of its area covered by fixed macroblocks or fixed cells. Area density constraints are presented in column 11.

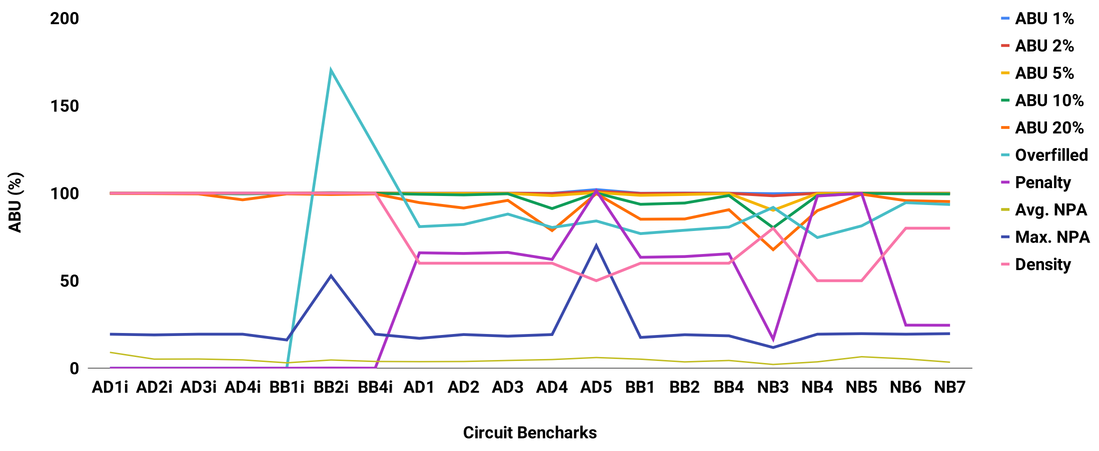
Table 8.5: ABU improvement from the proposed NFCS algorithm in aligned ABU and cell spreading grid graphs. ABU, NPA and area density restriction (density) are presented in percentage.

Circuits	ABU (%)							Avg. NPA	Max. NPA	Density (%)
	1%	2%	5%	10%	20%	OF	Penalty			
AD1i	100	100	100	100	100	0	0	9	19	100
AD2i	100	100	100	100	100	0	0	5	19	100
AD3i	100	100	100	100	100	0	0	5	19	100
AD4i	100	100	100	100	96	0	0	5	19	100
BB1i	100	100	100	100	100	0	0	3	16	100
BB2i	100	100	100	100	99	170	0	5	53	100
BB4i	100	100	100	100	100	126	0	4	19	100
AD1	100	100	100	99	95	81	66	4	17	60
AD2	100	100	100	99	92	82	66	4	19	60
AD3	100	100	100	100	96	88	66	4	18	60
AD4	100	100	99	91	79	80	62	5	19	60
AD5	102	101	100	100	100	84	101	6	70	50
BB1	100	100	99	94	85	77	63	5	18	60
BB2	100	100	99	94	85	79	64	4	19	60
BB4	100	100	100	99	91	81	65	4	19	60
NB3	100	99	90	80	68	92	17	2	12	80
NB4	100	100	100	99	90	75	98	4	19	50
NB5	100	100	100	100	99	81	100	7	20	50
NB6	100	100	100	100	96	95	25	5	19	80
NB7	100	100	100	100	95	94	25	3	20	80
Avg.	100	100	99	98	93	69	41	5	23	76

In Table 7.4 from Chapter NFL, ABU of the global placement solution with FastPlace is presented. In this experiment, area density restriction is set to 100% of bin's area for all circuits. ABU results achieved with the proposed NFCS algorithm are compared with ABU from global placement solutions. On average, in ABU 1%, ABU 2%, ABU 5% and ABU 10%, the area density utilization has been reduced equal to or lower than 100%. ABU of the proposed NFCS algorithm is compared with global placement solution (Table 7.4 from Chapter NFL). ABU of OF bins and ABU penalty have been minimized by 28% and 45%, on average, respectively. ABU is computed using the area density constraint established in the 2006 ISPD contest.

In Figure 8.5, ABU distributions of the proposed NFCS algorithm are presented. In abscissa, circuit acronyms are given. In ordinate, ABU in percentage is presented. In this figure, ABU 1%, ABU 2%, ABU 5%, ABU 10%, ABU 20%, ABU of OF bins (Overfilled), ABU penalty (Penalty), average NPA (Avg. NPA), maximum NPA (Max. NPA), and global placement maximum area density utilization constraints (Density) are shown. ABU metrics are evaluated only for valid ABU bins.

Figure 8.5: ABU distributions from the 2006 ISPD circuits are presented. Cell spreading and ABU grid graphs are aligned. Circuits have been placed with the FastPlace global placement algorithm



Source: Author (2019).

In Table 8.6, ACD of the proposed NFCS algorithm in aligned cell spreading and ABU grid graphs is presented. Cell displacement is presented in the number of row heights. In column 1, circuit acronyms are presented. ACD of cells with the highest cell displacements are given in columns 2 to 5. In columns 6 and 7, average and maximum cell displacement are shown, respectively. The number of moved cells and the total number of cells are given in columns 8 and 9, respectively. In column 10, the percentage of moved cells are presented.

On average, the ACD 2% has displacement of 50 row heights. The average and maximum cell displacement are 2.1 and 194 row heights, on average, respectively. On average, 12% of all movable cells are moved to minimize area density violations.

In Table 8.7, legalization results of Jezz and CSJ flows in aligned grid graphs are presented. In column 1, circuit acronyms (BK) are presented. HPWL of global placement solutions are given in column 2. Increased HPWL in the percentage of CSJ and Jezz flows are presented in columns 3 and 4, respectively. In columns 5 to 8, average and maximum cell displacements for Jezz and CSJ flows are shown. The runtime of Jezz and CSJ flows are highlighted in columns 9 and 10, respectively. In columns 11 and 12, the runtime of cell spreading and legalization stages of the proposed NFCS are presented, respectively.

The CSJ flow achieves 23% improvement on HPWL compared to Jezz, on average. In average and maximum cell displacement, CSJ flow minimizes cell displacement by 18% and 12% compared to Jezz, on average, respectively. Achieved improvement is

Table 8.6: ACD of the proposed NFCS algorithm in aligned ABU and cell spreading grid graphs. ACD is presented in number of row heights.

Circuits	ACD (Row Heights)				Avg. Disp.	Max. Disp.	Cells ($\times 10^3$)		%
	2%	5%	10%	20%			Moved	Total	
AD1i	10.8	6.7	4.5	2.8	0.6	51.0	51	211	24.1
AD2i	41.9	18.2	9.6	4.9	1.0	195.4	35	254	13.9
AD3i	139.4	83.9	49.7	27.3	5.6	446.2	151	451	33.6
AD4i	4.7	2.6	1.5	0.8	0.2	86.8	44	495	8.9
BB1i	3.7	2.2	1.3	0.6	0.1	73.0	24	278	8.5
BB2i	4.1	2.3	1.3	0.7	0.1	54.0	45	535	8.4
BB4i	3.5	2.0	1.1	0.5	0.1	124.0	158	2,169	7.3
AD1	2.5	1.3	0.6	0.3	0.1	45.7	8	211	3.9
AD2	86.9	43.9	22.2	11.1	2.2	197.3	21	254	8.3
AD3	151.3	83.9	47.1	24.7	4.9	400.7	98	451	21.7
AD4	29.6	13.3	6.6	3.3	0.7	136.6	28	495	5.7
AD5	276.3	203.0	142.4	85.8	17.9	676.4	320	842	37.9
BB1	4.8	1.9	1.0	0.5	0.1	78.3	5	278	2.0
BB2	8.7	3.5	1.8	0.9	0.2	51.4	14	535	2.5
BB4	25.4	12.2	6.2	3.1	0.6	194.0	153	2,169	7.1
NB3	0.7	0.3	0.1	0.1	0.0	26.0	6	483	1.3
NB4	24.2	11.1	5.6	2.8	0.6	111.9	44	643	6.8
NB5	122.4	79.4	49.7	27.6	5.6	419.8	318	1,228	25.9
NB6	34.0	16.1	8.4	4.2	0.8	225.4	126	1,248	10.1
NB7	24.0	10.2	5.1	2.6	0.5	280.5	138	2,481	5.6
Avg.	49.9	29.9	18.3	10.2	2.1	193.7	89	786	12.2

obtained by opening white space in high-density regions. Therefore, Jezz in CSJ flow effectively explores these newly opened space.

8.3.2.3 Experimental Results in benchmarks from the 2014 and 2015 ICCAD contests

In Table 8.8, experimental results from the ICCAD 2015 benchmarks are presented. Cell spreading and ABU grid graphs are not aligned. In column 1, circuit acronyms (BK) are presented. Initial and increased legalized-wire length in percentage (WL) for Jezz and CSJ flows are presented in columns 2 to 4. In columns 5 to 8, the average and maximum cell displacement of the legalized circuit for Jezz and CSJ flows are presented. Initial ABU is shown in column 9. In columns 10 to 15, ABU, ABU 2% and ABU 5% for Jezz and CSJ flows are given. The runtime (Time) of the cell spreading stage is presented in column 16.

The CSJ flow achieves 12% improvement on HPWL compared with Jezz, on average. Average and maximum cell displacements are improved by 7.4% and 46% in CSJ

Table 8.7: Legalization results from Jezz and CSJ flows in aligned ABU and cell spreading grid graphs.

BK	HPWL (m)	HPWL (%)		Avg. Disp. #Rows		Max. Disp. #Rows		Runtime (s)			
								Jezz	NFCS		
		Jezz	CSJ	Jezz	CSJ	Jezz	CSJ		Total	CS	Legal
AD1i	92	14.6	12.0	3.6	3.4	423	343	8.4	22.7	15.2	7.5
AD2i	108	8.9	8.6	3.5	2.9	366	557	6.4	22.4	15.1	7.2
AD3i	241	27.7	23.4	10.1	9.0	1032	1015	51.0	139.7	115.9	23.8
AD4i	225	4.9	4.4	1.7	1.6	208	158	8.8	20.0	11.6	8.4
BB1i	118	5.4	4.6	1.6	1.5	209	197	5.7	13.6	8.1	5.5
BB2i	179	6.9	6.3	1.8	1.7	284	269	9.7	22.9	13.6	9.3
BB4i	940	4.3	4.0	1.4	1.4	231	206	39.9	94.5	55.6	38.9
AD1	93	3.0	3.0	1.1	1.1	74	65	1.7	3.4	1.8	1.6
AD2	101	5.4	6.7	5.1	3.6	316	453	2.6	12.3	8.8	3.5
AD3	224	19.0	14.1	9.4	7.4	823	672	32.6	69.8	57.0	12.8
AD4	214	4.4	4.1	1.8	1.7	298	285	5.4	14.7	9.9	4.8
AD5	377	83.1	58.1	29.1	22.2	1078	950	283.3	812.6	731.7	80.9
BB1	112	2.4	2.4	0.9	0.9	143	128	2.0	3.4	1.5	1.9
BB2	170	4.1	3.4	1.2	1.1	232	110	6.1	13.2	7.7	5.5
BB4	898	5.5	4.8	1.7	1.7	376	373	30.3	129.1	102.0	27.1
NB3	302	1.5	1.5	0.8	0.8	52	38	4.0	6.5	2.7	3.8
NB4	288	5.0	4.4	1.8	1.7	250	232	9.3	34.6	27.2	7.3
NB5	476	34.3	28.1	8.9	8.2	753	644	89.0	482.6	438.7	43.9
NB6	530	8.5	7.0	2.2	2.2	522	500	27.5	145.5	119.6	25.9
NB7	1113	4.5	4.6	1.7	1.6	931	515	39.0	112.4	78.2	34.3
Norm.	-	1.23	1.00	1.18	1.00	1.12	1.00	0.30	1.00	0.84	0.16

compared with Jezz, on average, respectively. In both CSJ and Jezz, ABU is reduced 23%, on average.

In Table 8.9, two flows with and without the proposed algorithm before MDP (LIN et al., 2016) are evaluated. In MDP, only the binary of (LIN et al., 2016) is run while, in CS, the proposed NFCS algorithm is executed before MDP. This experiment was conducted in the modified circuits from the 2014 ICCAD benchmarks. In columns 1, circuit acronyms (BK) are presented. In columns 2 to 4, initial, MDP, and CS wire lengths (HPWL) are presented, respectively. In columns 5 and 6, the average cell displacement for MDP and CS flows are presented. The initial ABU is presented in column 7. In columns, 8 to 15, ABU, ABU 2%, ABU 5%, and ABU penalty for MDP and CS flows are presented. The runtime (Time) of the cell spreading stage is presented in column 16.

MDP and the proposed NFCS algorithm have optimization objectives which conflict with each other. The proposed NFCS algorithm minimizes high-density regions subject to minimum cell displacement, which can increase wire length. On the other hand, MDP focus on minimizing wire length with a relaxed restriction to area density utilization. Therefore, area spaces which were opened by NFCS algorithm are used by MDP to improve wire length. Improvements achieved individually by MDP and NFCS algorithms are degraded when both algorithms are executed in the same optimization flow.

8.4 Summary

In commercial and academic flows, the proposed NFCS algorithm moves cells out from high-density regions to open area spaces with minimized cell displacement cost. Opened spaces are available for detailed placement optimization algorithms to improve the quality of the circuit solution. In high-density regions, optimization algorithms can use spaces that have been opened previously by using the NFCS algorithm to place cells. In this approach, the placement solution can be improved. The outcome of the proposed NFCS algorithm is a placement solution with optimized area density utilization.

A cell spreading algorithm based on branch and cut, and network flow techniques is presented. The proposed cell spreading algorithm is applied in the detailed placement to optimize area density utilization. Cell movements are subject to maximum cell displacement constraint. Within the proposed NFCS algorithm, area spaces are opened in high-density regions. Detailed placement algorithms may explore these newly opened spaces. Notably, timing-driven algorithms can improve signal delay by exploring these spaces opened by the proposed NFCS algorithm. The proposed algorithm searches optimized-cost paths. In the grid graph, bins are opened only if ancestor bins have the displacement cost lower than the upper limit cost. In cell displacement cost model, the signal of cost value indicates the direction of cell movements. Positive cost value indicates that cells are going to be moved away from initial positions. These cell movements increase cell displacement. On the other hand, negative cost value indicates that cells are going to be moved closer to initial positions. Cell movements closer to initial positions lead to minimize cell displacement cost. In commercial circuits, improvement in timing and power consumption are achieved when the NFCS algorithm is used. Moreover, the improvement of wire length and area density utilization is achieved in academic and commercial circuits. In academic circuits, the proposed NFCS algorithm contributes to improve wire length and area density utilization that improves placement solution.

Table 8.8: Experimental results on academic circuits. The proposed NFCS algorithm is evaluated in two legalization flows. The first flow is composed only of the legalizer while the second flow is composed of the proposed algorithm and the legalizer.

BK	WL Init.	WL (%)		Avg. Disp.		Max. Disp.		ABU Init.	ABU		ABU 2%		ABU 5%		Time (s)
		Jezz	CSJ	Jezz	CSJ	Jezz	CSJ		Jezz	CSJ	Jezz	CSJ	Jezz	CSJ	
SB1	92.34	0.75	**0.74**	0.79	**0.78**	324	**248**	0.94	78.99	**78.56**	82.40	**81.82**	76.52	**76.26**	3.34
SB3	91.19	2.29	**2.24**	1.67	**1.63**	780	**288**	1.23	**99.98**	99.99	100.00	100.00	100.00	100.00	2.22
SB4	68.03	1.01	**0.84**	1.11	**1.01**	435	**326**	1.13	87.55	**87.36**	90.63	**90.39**	85.56	**85.42**	2.70
SB5	102.80	0.81	**0.60**	1.01	**0.87**	619	**299**	0.92	78.28	**77.55**	83.51	**82.52**	74.32	**73.83**	2.76
SB7	136.96	1.04	**1.00**	0.89	**0.87**	329	**308**	1.13	**85.06**	84.85	88.77	**88.48**	82.28	**82.17**	5.18
SB10	176.72	0.63	**0.44**	0.82	**0.72**	502	**490**	0.77	71.37	**70.74**	74.09	**73.25**	69.52	**69.10**	2.98
SB16	87.07	0.89	**0.69**	0.96	**0.84**	295	**272**	0.92	84.24	**83.67**	86.57	**85.81**	82.64	**82.24**	2.67
SB18	57.19	0.80	**0.74**	0.81	**0.77**	253	**191**	1.02	76.96	**76.26**	80.69	**79.74**	74.18	**73.74**	1.12
Avg.	101.54	1.03	**0.91**	1.01	**0.94**	442	**303**	1.01	82.80	**82.37**	85.83	**85.25**	80.63	**80.34**	2.87

Table 8.9: Experimental results of the proposed algorithm in academic benchmarks. The proposed NFCS algorithm is evaluated in two flows. In one flow, MDP (LIN et al., 2016) binary is executed before NFCS algorithm. In the second flow, only MDP is executed.

BK	WL (um) Init.	WL (%)		Avg. Disp.		ABU Init.	ABU		ABU 2%		ABU 5%		ABU P.		Time (s)
		MDP	CS	MDP	CS		MDP	CS	MDP	CS	MDP	CS	MDP	CS	
b19	3.32	3.14	3.14	1.87	1.87	79.28	**77.26**	77.30	**77.76**	77.82	**77.02**	77.04	**1.71**	1.76	0.14
dist	5.06	4.82	4.82	2.16	**2.15**	72.90	71.86	**71.66**	72.16	**71.92**	71.67	**71.49**	0.00	0.00	0.05
leon2	31.97	31.22	31.22	2.07	2.07	74.16	72.95	**72.91**	73.12	**73.07**	72.87	**72.82**	4.21	**4.15**	0.56
leon3mp	15.19	14.30	14.30	1.82	1.82	72.63	70.22	**70.18**	70.44	**70.39**	70.08	**70.05**	0.39	**0.34**	0.42
matrix	2.95	2.75	2.75	1.50	1.50	67.94	**66.82**	66.85	**67.09**	67.14	**66.66**	66.67	**2.80**	2.84	0.05
netcard	41.13	40.25	40.24	1.53	1.53	75.77	**74.81**	74.82	**75.12**	75.12	**74.59**	74.60	**3.91**	3.92	0.71
vga lcd	4.19	4.02	4.02	1.61	**1.60**	73.29	72.31	**72.18**	72.51	**72.35**	72.20	**72.09**	3.31	**3.11**	0.06
Avg.	14.83	14.36	14.36	1.79	1.79	73.71	72.32	**72.27**	72.60	**72.55**	72.15	**72.11**	2.33	**2.30**	0.28

9 CONCLUSIONS

In this thesis, two contributions are presented. The first contribution is an incremental timing-driven placement algorithm to optimize timing violations subject to routability. The second contribution is a generic cell spreading algorithm to move cells out from high-density regions with minimized adverse side effects on moved cells. The proposed cell spreading algorithm relies on network flow and branch and cut formal methods to provide optimized placement solutions. A cost model is integrated into the proposed cell spreading algorithm that can quickly provide the history and direction of cell movements. The proposed cell spreading technique is applied in legalization and detailed placement stages.

In RAITDP flow, a simplified cell spreading stage that minimizes area density violation is used. However, this approach may not handle regions with high cell concentration or high-density regions, which are very large. Moreover, the maximum cell displacement restriction limits the distance to move cells out from high-density regions. The RAITDP flow can benefit from the proposed network flow-based cell spreading algorithm. Cells could be moved out from high-density regions subject to maximum cell displacement constraint.

The traditional placement flow (global placement, legalization, and detailed placement) can be restrictive to optimize placed netlist in detailed placement. Moreover, legalization algorithms may cause adverse side effects in the placement solution. Legalization algorithms' primary focus is to fix cell overlapping instead of moving cells out from high-density regions. On the other hand, detailed placement algorithms may further explore the optimization space to provide improved placement solutions if legalization restriction is relaxed. Placement solution can be improved if legalization and detailed placement flows could be interleaved. The main issue of this mixed flow is cell concentration generated by detailed placement optimization algorithms. In traditional flow, legalization algorithms will spread these cells far away. Therefore, achieved placement improvement could be reverted. An algorithm which can spread cells with minimized adverse side effects can mitigate drawbacks on placement solution after legalization.

The proposed network flow-based cell spreading algorithm can fix area density violation with minimized side effects. In the proposed algorithm, restrictions such as timing, routing overflow, and power consumption, can be addressed when cells are selected to be moved out from high-density regions. The proposed algorithm can be inserted in several stages of mixed legalization and detailed placement flow. The cell spreading algorithm can adequately minimize area density violations generated by detailed placement algorithms. The legalization algorithm can also benefit from the placement solution provided by the proposed cell spreading algorithm. The legalization stage will require less effort to provide optimized placement solution. Adverse side effects are also minimized on the legal placement solution.

9.1 Summary of Contributions

The summary of thesis contributions are listed as follows:

- Designing a detailed placement flow to move critical timing cells to optimized-timing positions subject to routabiliy. Cell movements are accepted only if target positions have a lower routing violation than the current position or target positions are free of routing violations. Cell displacement constraint is evaluated to determine the trade-off between cell displacement restriction and improvements on timing violations. Routing overflow aspects are evaluated by reducing the available routing resources.
- A cell spreading algorithm with network flow and branch and cut techniques is proposed. The objective is to optimize area density violation with minimized adverse side effects on moved cells. Optimized cost paths are computed from the overfilled bin. Cells are moved between neighboring bins to provide area flow to be moved out from high-density regions. Optimized cost paths are computed with network flow and branch and cut techniques. Branch and cut technique is used to restrict the search in the tree to open only branches which have the cost path lower than the upper limit cost. The upper limit cost is iteratively reduced every time a new path with lower cost is found. This path that has the cost lower than the upper limit establishes the new reduced upper limit cost. The algorithm stops to augment paths as soon as there are no more bins which can be opened with ancestor cost lower than the upper limit. Therefore, the path with the lowest cost is obtained. A cost model with the history and direction of cell movements is integrated in the proposed cell spreading technique. Cells which are placed inside or close to regions with the high-density area can be moved in several paths. These cells can be moved to any direction related to initial positions. In some paths, these cells can be moved back to initial

positions. On the other side, in other paths, these cells can be moved distant from initial positions. The direction of cell movement can significantly impact the total cell displacement cost. In the cost model of the proposed network flow-based cell spreading, the signal of the cost (positive or negative) indicates the direction of cell movement. Negative cost values indicate cell movements closer to initial positions. Otherwise, cell movements distant from initial positions have positive cost values. In the proposed cell spreading algorithm, cells are ranked by the lowest to the highest costs. Cells are selected from the lowest to the highest cost while the required area flow to be moved out is not achieved. Paths can have sets of cell movements between neighbor bins with negative cost values. These negative values indicate cells will be moved closer to initial positions that minimize cell displacement cost.

- In legalization, the proposed network flow-based cell spreading technique is applied to move cells out of high-density regions. The proposed algorithm reduces cell concentration in high-density regions. Therefore, legal spaces to place cells are found with minimized cell displacement. The legalization flow is composed of the cell spreading and legalization stage. In the cell spreading stage, cells are moved out of high-density regions with the proposed cell spreading algorithm. In the legalization stage, cells are placed in legal positions with Jezz (that is an abacus-based) algorithm. The total cell displacement of legal placement solution is improved with the proposed cell spreading algorithm.
- In detailed placement, the proposed network flow-based cell spreading technique is applied to move cells out from high-density regions. In high-density regions, opened white spaces with the proposed algorithm may be used by optimization algorithms to improve wire length, power consumption, timing violations, and routability. The proposed cell spreading algorithm opens white space in high-density regions with minimized adverse side effects. The opened spaces are available to be used by optimization detailed placement algorithms. Therefore, optimization algorithms may further improve the placement solution by placing cells in these newly opened positions.

9.2 Future Works

In the network flow-based legalization, the proposed algorithm can be extended to legalize multi-deck cells. However, graph grid and path augmentation must be redesign to address cells which cover several rows. Multi-deck cells are very challenging to legalize because these cells require white space in several rows. The white space must be consecutive in adjacent rows. Each multi-deck cell movement may require to move cells in several rows to open necessary space. Another challenge is that rows will have different heights. Therefore, some standard-cells will be possible to place in a row, but these cells will not fit in neighboring rows because of the difference in row height.

In the network flow-based cell spreading, the proposed algorithm can be designed to move cells to neighbor bins subject to keeping circuit legalized. In the proposed cell spreading algorithm, the optimized placement solution is illegal. This placement solution must be legalized after optimizing area density with the proposed algorithm. This approach can have some adverse side effects because of the legalization process. On the other hand, these adverse side effects can be avoided by moving cells to legal positions. In this approach, the challenge is to compute cells to be moved considering legal positions. A legal position can be inside of white spaces or overlapping cells which will be moved to a neighboring bin. Moreover, computing area flow to be moved out considering legal positions is more restrictive than computing area flow in an illegal placement. Both versions of detailed placement cell spreading algorithms can be integrated into the mixed legalization and detailed placement flow.

The network flow-based cell spreading algorithm can be integrated into the analytical global placement flow to provide the upper bound placement solution. In the global placement, lower and upper bound placement solutions are obtained by solving a system of linear equations and by spreading cells in the circuit core, respectively. In the lower bound placement solution, cell concentration is very high in a considerable number of small regions. Therefore, these cells must be spread to regions with low-density area utilization. In the conventional upper bound placement solution, cells can have a significant placement displacement, which negatively affects the optimality of the global placement solution.

9.3 Publications, Awards, and Open-Source Framework

9.3.1 Journal Publication

1. **J. Monteiro**, M. Johann and L. Behjat, "*An Optimized Cost Flow Algorithm to Spread Cells in Detailed Placement*", ACM Transactions on Design Automation of Electronic Systems, 2019.

9.3.2 Conferences, Symposiums and Workshops

1. G. Flach, M. Fogaça, **J. Monteiro**, M. Johann, and R. Reis, "*Rsyn: An Extensible Physical Synthesis Framework*", In Proceedings of the 2017 ACM on International Symposium on Physical Design (ISPD '17), ACM, New York, NY, USA, 33-40.
2. M. Fogaça, G. Flach, **J. Monteiro**, M. Johann and R. Reis, "*Quadratic timing objectives for incremental timing-driven placement optimization*", 2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS), Monte Carlo, 2016, pp. 620-623.
3. **J. Monteiro**, N. K. Darav, G. Flach, M. Fogaça, R. Reis, A. Kennings, M. Johann, L. Behjat, "*Routing-Aware Incremental Timing-Driven Placement*", 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Pittsburgh, PA, 2016, pp. 290-295.
4. G. Flach, M. Fogaça, **J. Monteiro**, M. Johann, and R. Reis, "*Drive Strength Aware Cell Movement Techniques for Timing Driven Placement*", In Proceedings of the 2016 on International Symposium on Physical Design (ISPD '16), ACM, New York, NY, USA, 73-80.
5. G. Flach, **J. Monteiro**, M. Fogaça, J. Puget, P. Butzen, M. Johann and R. Reis, "*An Incremental Timing-Driven flow using quadratic formulation for detailed placement*", 2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Daejeon, 2015, pp. 1-6.
6. **J. Monteiro**, G. Flach, M. Johann and J. L. A. Güntzel, "*An analytical timing-driven algorithm for detailed placement*", 2015 IEEE 6th Latin American Symposium on Circuits & Systems (LASCAS), Montevideo, 2015, pp. 1-4.

9.3.3 Awards

1. 2019 Association for Computing Machinery (ACM) Ph.D. Forum at Design Automation Conference (DAC).
2. 2018 ISPD contest on Initial Detailed Routing. *4th place*.
3. 2016 Emerging Leaders of the Americas Program (ELAP) to attend the University of Calgary as visiting researcher student.
4. 2016 A. Richard Newton Young Student Fellow.
5. 2015 ICCAD contest on Incremental Timing-driven Placement. *2nd place*.
6. 2014 ICCAD contest on Incremental Timing-driven Placement. *1st place*.

9.3.4 Open-source Framework in EDA

- *RsynDesign Open-source Framework* available at GitHub <https://github.com/RsynTeam> repository.

REFERENCES

ACHTERBERG, T.; KOCH, T.; MARTIN, A. Branching rules revisited. **Operations Research Letters**, v. 33, n. 1, p. 42 – 54, 2005. ISSN 0167-6377. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0167637704000501>.

ACM/SIGDA benchmarks. 2017. Available from Internet: <https://people.engr.ncsu.edu/brglez/CBL/benchmarks/Benchmarks-upto-1996.html>.

AHUJA, R. K.; MAGNANTI, T. L.; ORLIN, J. B. **Network Flows: Theory, Algorithms, and Applications**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993. ISBN 0-13-617549-X.

ALPERT, C. et al. Placement: Hot or not? In: **Proceedings of ICCAD**. [S.l.: s.n.], 2012. p. 283–290. ISSN 1092-3152.

ALPERT, C. J.; MEHTA, D. P.; SAPATNEKAR, S. S. **Handbook of Algorithms for Physical Design Automation**. 1st. ed. Boston, MA, USA: Auerbach Publications, 2008. ISBN 0849372429, 9780849372421.

BOCK, A. et al. Local search algorithms for timing-driven placement under arbitrary delay models. In: **Proceedings of DAC**. New York, NY, USA: ACM, 2015. (DAC ’15), p. 29:1–29:6. ISBN 978-1-4503-3520-1. Available from Internet: <http://doi.acm.org/10.1145/2744769.2744867>.

BONDY, J. A. **Graph Theory With Applications**. Oxford, UK, UK: Elsevier Science Ltd., 1976. ISBN 0444194517.

BOOKSHELF. 2017. Available from Internet: <http://vlsicad.eecs.umich.edu/BK/ISPD06bench/BookshelfFormat.txt>.

BORKAR, S. Design perspectives on 22nm cmos and beyond. In: **2009 46th ACM/IEEE Design Automation Conference**. [S.l.: s.n.], 2009. p. 93–94. ISSN 0738-100X.

BORKAR, S. et al. Parameter variations and impact on circuits and microarchitecture. In: **Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)**. [S.l.: s.n.], 2003. p. 338–342.

BRENNER, U. Vlsi legalization with minimum perturbation by iterative augmentation. In: **Proceedings of the Conference on Design, Automation and Test in Europe**. San Jose, CA, USA: EDA Consortium, 2012. (DATE ’12), p. 1385–1390. ISBN 978-3-9810801-8-6.

BRENNER, U. Bonnplace legalization: Minimizing movement by iterative augmentation. **IEEE Trans. on CAD**, v. 32, n. 8, p. 1215–1227, Aug 2013.

BRENNER, U.; STRUZYNA, M.; VYGEN, J. Bonnplace: Placement of leading-edge chips by advanced combinatorial algorithms. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 27, n. 9, p. 1607–1620, Sept 2008. ISSN 0278-0070.

BURSTEIN, M.; YOUSSEF, M. N. Timing influenced layout design. In: **Proceedings of DAC**. [S.l.: s.n.], 1985. p. 124–130. ISSN 0738-100X.

CALDWELL, A. E.; KAHNG, A. B.; MARKOV, I. L. Can recursive bisection alone produce routable placements? In: **Proceedings of DAC**. New York, NY, USA: ACM, 2000. p. 477–482. ISBN 1-58113-187-9.

CAULEY, S. et al. A parallel branch-and-cut approach for detailed placement. **ACM Trans. Des. Autom. Electron. Syst.**, ACM, New York, NY, USA, v. 16, n. 2, p. 18:1–18:19, abr. 2011. ISSN 1084-4309.

CHAN, T. F. et al. Multilevel optimization for large-scale circuit placement. In: **IEEE/ACM International Conference on Computer Aided Design. ICCAD - 2000. IEEE/ACM Digest of Technical Papers (Cat. No.00CH37140)**. [S.l.: s.n.], 2000. p. 171–176. ISSN 1092-3152.

CHAN, T. F. et al. mpl6: Enhanced multilevel mixed-size placement. In: **Proceedings of the 2006 International Symposium on Physical Design**. New York, NY, USA: ACM, 2006. (ISPD '06), p. 212–214. ISBN 1-59593-299-2.

CHANG, C.-C.; CONG, J.; PAN, Z. D. Physical hierarchy generation with routing congestion control. In: **Proceedings of the 2002 International Symposium on Physical Design**. New York, NY, USA: ACM, 2002. (ISPD '02), p. 36–41. ISBN 1-58113-460-6.

CHANG, C.-C. et al. Optimality and scalability study of existing placement algorithms. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 23, n. 4, p. 537–549, April 2004. ISSN 0278-0070.

CHEN, T. C. et al. Ntuplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 27, n. 7, p. 1228–1240, July 2008. ISSN 0278-0070.

CHENG, C. et al. Replace: Advancing solution quality and routability validation in global placement. **IEEE Trans. on CAD**, p. 1–1, 2018. ISSN 0278-0070.

CHO, M. et al. History-based vlsi legalization using network flow. In: **Proceedings of DAC**. [S.l.: s.n.], 2010. p. 286–291.

CHOU, S.; HO, T. Y. Oal: An obstacle-aware legalization in standard cell placement with displacement minimization. In: **2009 IEEE International SOC Conference (SOCC)**. [S.l.: s.n.], 2009. p. 329–332. ISSN 2164-1676.

CHOW, W.-K. et al. Cell density-driven detailed placement with displacement constraint. In: **Proceedings of ISPD**. New York, NY, USA: ACM, 2014. p. 3–10. ISBN 978-1-4503-2592-9.

CHOW, W.-K.; PUI, C.-W.; YOUNG, E. F. Y. Legalization algorithm for multiple-row height standard cell design. In: **Proceedings of DAC**. New York, NY, USA: ACM, 2016. p. 83:1–83:6. ISBN 978-1-4503-4236-0.

CHU, C.; WONG, Y. C. Flute: Fast lookup table based rectilinear steiner minimal tree algorithm for vlsi design. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 27, n. 1, p. 70–83, Jan 2008. ISSN 0278-0070.

CHUNG, E.-Y. et al. Advanced delay analysis method for submicron asic technology. In: **[1992] Proceedings. Fifth Annual IEEE International ASIC Conference and Exhibit**. [S.l.: s.n.], 1992. p. 471–474.

CONG, J. et al. Optimizing routability in large-scale mixed-size placement. In: **2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC)**. [S.l.: s.n.], 2013. p. 441–446. ISSN 2153-6961.

CONG, J.; XIE, M. A robust detailed placement for mixed-size ic designs. In: **Asia and South Pacific Conference on Design Automation, 2006.** [S.l.: s.n.], 2006. p. 7 pp.–. ISSN 2153-6961.

CONG, J.; XIE, M. A robust mixed-size legalization and detailed placement algorithm. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 27, n. 8, p. 1349–1362, Aug 2008. ISSN 0278-0070.

CORMEN, T. H. et al. **Introduction to Algorithms, Third Edition**. 3rd. ed. [S.l.]: The MIT Press, 2009. ISBN 0262033844, 9780262033848.

DARAV, N. K. et al. A fast, robust network flow-based standard-cell legalization method for minimizing maximum movement. In: **Proceedings of ISPD**. [S.l.: s.n.], 2017. p. 141–148. ISBN 978-1-4503-4696-2.

DARAV, N. K. et al. The impact of industry-organized contests on eda education. In: **2015 IEEE International Conference on Microelectronics Systems Education (MSE)**. [S.l.: s.n.], 2015. p. 21–24.

DARAV, N. K. et al. Eh?placer: A high-performance modern technology-driven placer. **Trans. of ACM TODAES**, v. 21, n. 3, abr. 2016.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. **Numerische Mathematik**, v. 1, n. 1, p. 269–271, 1959. ISSN 0945-3245.

DINIC, E. A. Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation. **Soviet Math Doklady**, v. 11, p. 1277–1280, 1970.

DOBRE, S.; KAHNG, A. B.; LI, J. Mixed cell-height implementation for improved design quality in advanced nodes. In: **2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)**. [S.l.: s.n.], 2015. p. 854–860.

DOLL, K.; JOHANNES, F. M.; ANTREICH, K. J. Iterative placement improvement by network flow methods. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 13, n. 10, p. 1189–1200, Oct 1994. ISSN 0278-0070.

DOLL, K.; JOHANNES, F. M.; SIGL, G. DOMINO: deterministic placement improvement with hill-climbing capabilities. In: **VLSI**. [S.l.: s.n.], 1991. p. 91–100.

DU, Y.; WONG, M. D. F. Optimization of standard cell based detailed placement for 16 nm finfet process. In: **Proceedings of the Conference on Design, Automation & Test in Europe**. 3001 Leuven, Belgium, Belgium: European Design and Automation Association, 2014. (DATE '14), p. 357:1–357:6. ISBN 978-3-9815370-2-4.

EDMONDS, J.; KARP, R. M. Theoretical improvements in algorithmic efficiency for network flow problems. **J. ACM**, ACM, New York, NY, USA, v. 19, n. 2, p. 248–264, abr. 1972. ISSN 0004-5411. Available from Internet: <http://doi.acm.org/10.1145/321694.321699>.

EISENMANN, H.; JOHANNES, F. M. Generic global placement and floorplanning. In: **Proceedings 1998 Design and Automation Conference. 35th DAC. (Cat. No.98CH36175)**. [S.l.: s.n.], 1998. p. 269–274.

ELMORE, W. C. The transient response of damped linear networks with particular regard to wideband amplifiers. **Journal of Applied Physics**, v. 19, n. 1, p. 55–63, 1948.

ENVISIA ultra placer reference, QPlace Version 5.1.55: Cadence Design Systems Inc. 1999.

EVEN, S. **Graph Algorithms**. 2nd. ed. New York, NY, USA: Cambridge University Press, 2011. ISBN 0521736536, 9780521736534.

FIDUCCIA, C. M.; MATTHEYSES, R. M. A linear-time heuristic for improving network partitions. In: **19th Design Automation Conference**. [S.l.: s.n.], 1982. p. 175–181. ISSN 0146-7123.

FLACH, G. et al. Drive strength aware cell movement techniques for timing driven placement. In: **Proceedings of ISPD**. New York, NY, USA: ACM, 2016. p. 73–80. ISBN 978-1-4503-4039-7.

FLACH, G. et al. Rsyn: An extensible physical synthesis framework. In: **Proceedings of ISPD**. [S.l.: s.n.], 2017. p. 33–40. ISBN 978-1-4503-4696-2.

FLACH, G. et al. An incremental timing-driven flow using quadratic formulation for detailed placement. In: **2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)**. [S.l.: s.n.], 2015. p. 1–6. ISSN 2324-8432.

FLACH, G. et al. Effective method for simultaneous gate sizing and v th assignment using lagrangian relaxation. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 33, n. 4, p. 546–557, April 2014. ISSN 0278-0070.

FLACH, G. A. **Discrete gate sizing and timing-driven detailed placement for the design of digital circuits**. Thesis (PhD) — Universidade Federal do Rio Grande do Sul, 12 2015.

FOGAçA, M. et al. Quadratic timing objectives for incremental timing-driven placement optimization. In: **ICECS**. [S.l.: s.n.], 2016. p. 620–623.

FORD, L. R.; FULKERSON, D. R. Maximal flow through a network. In: ____. **Classic Papers in Combinatorics**. Boston, MA: Birkhäuser Boston, 1987. p. 243–248. ISBN 978-0-8176-4842-8. Available from Internet: <https://doi.org/10.1007/978-0-8176-4842-8_15>.

GOMORY, R. E. Outline of an algorithm for integer solutions to linear programs. **Bull. Amer. Math. Soc.**, American Mathematical Society, v. 64, n. 5, p. 275–278, 09 1958.

GUTH, C. et al. Timing-driven placement based on dynamic net-weighting for efficient slack histogram compression. In: **Proceedings of ISPD**. New York, NY, USA: ACM, 2015. p. 141–148. ISBN 978-1-4503-3399-3.

HAN, K.; KAHNG, A. B.; LEE, H. Scalable detailed placement legalization for complex sub-14nm constraints. In: **Proceedings of the IEEE/ACM International Conference on Computer-Aided Design**. Piscataway, NJ, USA: IEEE Press, 2015. (ICCAD '15), p. 867–873. ISBN 978-1-4673-8389-9.

HE, X. et al. Ripple 2.0: High quality routability-driven placement via global router integration. In: **DAC**. [S.l.: s.n.], 2013. p. 1–6. ISSN 0738-100X.

HE, X. et al. Ripple: An effective routability-driven placer by iterative cell movement. In: **2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)**. [S.l.: s.n.], 2011. p. 74–79. ISSN 1092-3152.

HE, X. et al. Ripple: A robust and effective routability-driven placer. **TCAD**, v. 32, n. 10, p. 1546–1556, Oct 2013. ISSN 0278-0070.

HELD, S.; SCHORR, U. Post-routing latch optimization for timing closure. In: **DAC**. [S.l.]: ACM, 2014. p. 7:1–7:6. ISBN 978-1-4503-2730-5.

HILL, D. **Method and system for high speed detailed placement of cells within an integrated circuit design**. [S.l.]: Google Patents, 2002. US Patent 6,370,673.

HO, T.-Y.; LIU, S.-H. Fast legalization for standard cell placement with simultaneous wirelength and displacement minimization. In: **2010 18th IEEE/IFIP International Conference on VLSI and System-on-Chip**. [S.l.: s.n.], 2010. p. 369–374. ISSN 2324-8432.

HOROWITZ, E.; SAHNI, S.; RAJASCKARAN, S. **Computer Algorithms: C++**. New York, NY, USA: W. H. Freeman & Co., 1996. ISBN 0716783150.

HSU, M. K. et al. Routability-driven analytical placement for mixed-size circuit designs. In: **2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)**. [S.l.: s.n.], 2011. p. 80–84. ISSN 1092-3152.

HU, B.; MAREK-SADOWSKA, M. Multilevel fixed-point-addition-based vlsi placement. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 24, n. 8, p. 1188–1203, Aug 2005. ISSN 0278-0070.

HU, J. et al. An effective legalization approach based on multiple ordering. In: **2013 International Conference on Communications, Circuits and Systems (ICCCAS)**. [S.l.: s.n.], 2013. v. 2, p. 514–518.

HU, S.; LI, Z.; ALPERT, C. J. A fully polynomial time approximation scheme for timing driven minimum cost buffer insertion. In: **Proceedings of the 46th Annual Design Automation Conference**. New York, NY, USA: ACM, 2009. (DAC '09), p. 424–429. ISBN 978-1-60558-497-3.

HUANG, C. C. et al. Detailed-routing-driven analytical standard-cell placement. In: **ASP-DAC**. [S.l.: s.n.], 2015. p. 378–383. ISSN 2153-6961.

HUANG, C. C. et al. Detailed-routability-driven analytical placement for mixed-size designs with technology and region constraints. In: **ICCAD**. [S.l.: s.n.], 2015. p. 508–513.

HUANG, C. C. et al. Timing-driven cell placement optimization for early slack histogram compression. In: **2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)**. [S.l.: s.n.], 2016. p. 1–6.

HUANG, T. W.; WONG, M. D. F. Opentimer: A high-performance timing analysis tool. In: **Proceedings on ICCAD**. [S.l.: s.n.], 2015. p. 895–902.

ICCAD - International Conference On Computer Aided Design. 2017. Available from Internet: <https://iccad.com/>.

INTEL Core i7. 2018. Available from Internet: <https://www.intel.com/content/www/us/en/products/processors/core/i7-processors.html>.

ISPD - International Symposium on Physical Design. 2017. Available from Internet: <http://www.ispd.cc/>.

JUNG, J. et al. Owaru: Free space-aware timing-driven incremental placement. In: **Proceedings of the 35th International Conference on Computer-Aided Design**. New York, NY, USA: ACM, 2016. (ICCAD '16), p. 8:1–8:8. ISBN 978-1-4503-4466-1.

KAHNG, A. B. et al. **VLSI Physical Design: From Graph Partitioning to Timing Closure**. 1st. ed. [S.l.]: Springer London, Limited, 2011. ISBN 9789048195916.

KAHNG, A. B.; WANG, Q. A faster implementation of aplace. In: **Proceedings of the 2006 International Symposium on Physical Design**. New York, NY, USA: ACM, 2006. (ISPD '06), p. 218–220. ISBN 1-59593-299-2.

KARYPIS, G.; KUMAR, V. A hypergraph partitioning package. **ACM Transactions on Architecture and Code Optimization - TACO**, 01 1998.

KENNINGS, A.; DARAV, N. K.; BEHJAT, L. Detailed placement accounting for technology constraints. In: **2014 22nd International Conference on Very Large Scale Integration (VLSI-SoC)**. [S.l.: s.n.], 2014. p. 1–6. ISSN 2324-8432.

KERNIGHAN, B. W.; LIN, S. An efficient heuristic procedure for partitioning graphs. **The Bell System Technical Journal**, v. 49, n. 2, p. 291–307, Feb 1970. ISSN 0005-8580.

KHACHATURYAN, A.; SEMENOVSOVSKAYA, S.; VAINSHTEIN, B. The thermodynamic approach to the structure analysis of crystals. **Acta Crystallographica Section A**, v. 37, n. 5, p. 742–754, Sep 1981.

KIM, M. C. et al. A simplr method for routability-driven placement. In: **ICCAD**. [S.l.: s.n.], 2011. p. 67–73. ISSN 1092-3152.

KIM, M. C. et al. Iccad-2015 cad contest in incremental timing-driven placement and benchmark suite. In: **Proceedings of ICCAD**. [S.l.: s.n.], 2015. p. 921–926.

KIM, M. C.; HUJ, J.; VISWANATHAN, N. Iccad-2014 cad contest in incremental timing-driven placement and benchmark suite: Special session paper: Cad contest. In: **ICCAD**. [S.l.: s.n.], 2014. p. 361–366. ISSN 1092-3152.

KIM, M. C.; LEE, D. J.; MARKOV, I. L. Simpl: An effective placement algorithm. **IEEE Trans. on CAD**, v. 31, n. 1, p. 50–60, Jan 2012. ISSN 0278-0070.

KIM, M. C.; MARKOV, I. L. Complx: A competitive primal-dual lagrange optimization for global placement. In: **DAC Design Automation Conference 2012**. [S.l.: s.n.], 2012. p. 747–755. ISSN 0738-100X.

KIM, M.-C. et al. Maple: Multilevel adaptive placement for mixed-size designs. In: **Proceedings of ISPD**. New York, NY, USA: ACM, 2012. p. 193–200. ISBN 978-1-4503-1167-0.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. **Science**, American Association for the Advancement of Science, v. 220, n. 4598, p. 671–680, 1983. ISSN 0036-8075.

KLEINHANS, J. M. et al. Gordian: Vlsi placement by quadratic programming and slicing optimization. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 10, n. 3, p. 356–365, Mar 1991. ISSN 0278-0070.

KONG, T. A novel net weighting algorithm for timing-driven placement. In: **Proceedings of ICCAD**. [S.l.: s.n.], 2002. p. 172–176. ISSN 1092-3152.

KUHN, H. W. The hungarian method for the assignment problem. **Naval Research Logistics Quarterly**, Wiley Subscription Services, Inc., A Wiley Company, v. 2, n. 1-2, p. 83–97, 1955.

KURD, N. A. et al. Westmere: A family of 32nm ia processors. In: **2010 IEEE International Solid-State Circuits Conference - (ISSCC)**. [S.l.: s.n.], 2010. p. 96–97. ISSN 2376-8606.

LAND, A. H.; DOIG, A. G. An automatic method of solving discrete programming problems. **Econometrica**, The Econometric Society, v. 28, n. 3, p. pp. 497–520, 1960. ISSN 00129682.

LANDMAN, B. S.; RUSSO, R. L. On a pin versus block relationship for partitions of logic graphs. **IEEE Transactions on Computers**, C-20, n. 12, p. 1469–1479, Dec 1971. ISSN 0018-9340.

LEE, Y. M.; WU, T.-Y.; CHIANG, P.-Y. A hierarchical bin-based legalizer for standard-cell designs with minimal disturbance. In: **2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)**. [S.l.: s.n.], 2010. p. 568–573. ISSN 2153-6961.

LI, S.; KOH, C.-K. Mixed integer programming models for detailed placement. In: **Proceedings of the 2012 ACM International Symposium on International Symposium on Physical Design**. New York, NY, USA: ACM, 2012. (ISPD ’12), p. 87–94. ISBN 978-1-4503-1167-0.

LI, S.; KOH, C.-k. Mip-based detailed placer for mixed-size circuits. In: **Proceedings of the 2014 on International Symposium on Physical Design**. New York, NY, USA: ACM, 2014. (ISPD ’14), p. 11–18. ISBN 978-1-4503-2592-9.

LIN, T.; CHU, C. Polar 2.0: An effective routability-driven placer. In: **2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)**. [S.l.: s.n.], 2014. p. 1–6. ISSN 0738-100X.

LIN, T. et al. Polar: Placement based on novel rough legalization and refinement. In: **2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)**. [S.l.: s.n.], 2013. p. 357–362. ISSN 1092-3152.

LIN, Y. et al. Mrdp: Multiple-row detailed placement of heterogeneous-sized cells for advanced nodes. In: **Proceedings of the 35th International Conference on Computer-Aided Design**. New York, NY, USA: ACM, 2016. (ICCAD '16), p. 7:1–7:8. ISBN 978-1-4503-4466-1.

LIU, W. H. et al. Nctu-gr 2.0: Multithreaded collision-aware global routing with bounded-length maze routing. **TCAD**, v. 32, n. 5, p. 709–722, May 2013. ISSN 0278-0070.

LIVRAMENTO, V. et al. Exploiting non-critical steiner tree branches for post-placement timing optimization. In: **Proceedings of the IEEE/ACM International Conference on Computer-Aided Design**. Piscataway, NJ, USA: IEEE Press, 2015. (ICCAD '15), p. 528–535. ISBN 978-1-4673-8389-9.

LIVRAMENTO, V. et al. Clock-tree-aware incremental timing-driven placement. **ACM Trans. Des. Autom. Electron. Syst.**, ACM, New York, NY, USA, v. 21, n. 3, p. 38:1–38:27, abr. 2016. ISSN 1084-4309.

LIVRAMENTO, V. S. et al. A hybrid technique for discrete gate sizing based on lagrangian relaxation. **ACM Trans. Des. Autom. Electron. Syst.**, ACM, New York, NY, USA, v. 19, n. 4, p. 40:1–40:25, aug. 2014. ISSN 1084-4309.

LU, J. et al. eplace: Electrostatics based placement using nesterov's method. In: **2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)**. [S.l.: s.n.], 2014. p. 1–6. ISSN 0738-100X.

LUO, T.; PAN, D. Z. Dplace2.0: A stable and efficient analytical placement based on diffusion. In: **2008 Asia and South Pacific Design Automation Conference**. [S.l.: s.n.], 2008. p. 346–351. ISSN 2153-6961.

MALHOTRA, V.; KUMAR, M.; MAHESHWARI, S. An o(|v|3) algorithm for finding maximum flows in networks. **Information Processing Letters**, v. 7, n. 6, p. 277 – 278, 1978. ISSN 0020-0190.

MARKOV, I. L.; HU, J.; KIM, M. C. Progress and challenges in vlsi placement research. **Proceedings of the IEEE**, v. 103, n. 11, p. 1985–2003, Nov 2015. ISSN 0018-9219.

MARTINS, M. et al. Open cell library in 15nm freepdk technology. In: **Proceedings of the 2015 Symposium on International Symposium on Physical Design**. New York, NY, USA: ACM, 2015. (ISPD '15), p. 171–178. ISBN 978-1-4503-3399-3.

MONTEIRO, J. et al. Routing-aware incremental timing-driven placement. In: **Proceedings of ISVLSI**. [S.l.: s.n.], 2016. p. 290–295.

MONTEIRO, J. et al. An analytical timing-driven algorithm for detailed placement. In: **2015 IEEE 6th Latin American Symposium on Circuits Systems (LASCAS)**. [S.l.: s.n.], 2015. p. 1–4.

MONTEIRO, J.; JOHANN, M.; BEHJAT, L. An optimized cost flow algorithm to spread cells in detailed placement. **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, ACM, New York, NY, USA, v. 24, n. 3, p. 35:1–35:16, abr. 2019. ISSN 1084-4309. Available from Internet: <http://doi.acm.org/10.1145/3317575>.

NAGEL, L. W.; PEDERSON, D. **SPICE (Simulation Program with Integrated Circuit Emphasis)**. [S.l.], 1973. Available from Internet: <http://www2.eecs.berkeley.edu/Pubs/TechRpts/1973/22871.html>.

NAM, G.; CONG, J. **Modern Circuit Placement: Best Practices and Results**. [S.l.]: Springer US, 2007. (Integrated Circuits and Systems). ISBN 9780387687391.

NANGATE FreePDK45 Generic Open Cell Library. 2017. Available from Internet: <http://projects.si2.org/openeda.si2.org/projects/nangatelib>.

NETTO, R. et al. Exploiting parallelism to speed up circuit legalization. In: **2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)**. [S.l.: s.n.], 2016. p. 624–627.

NETTO, R. et al. Speeding up incremental legalization with fast queries to multidimensional trees. In: **2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)**. [S.l.: s.n.], 2016. p. 36–41.

ODABASIOGLU, A.; CELIK, M.; PILEGGI, L. T. Prima: passive reduced-order interconnect macromodeling algorithm. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 17, n. 8, p. 645–654, Aug 1998. ISSN 0278-0070.

ORLIN, J. B. Max flows in o(nm) time, or better. In: **Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing**. New York, NY, USA: ACM, 2013. (STOC '13), p. 765–774. ISBN 978-1-4503-2029-0. Available from Internet: <http://doi.acm.org/10.1145/2488608.2488705>.

PADBERG, M.; RINALDI, G. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. **SIAM Rev.**, v. 33, n. 1, p. 60–100, feb. 1991.

PAN, M.; VISWANATHAN, N.; CHU, C. An efficient and effective detailed placement algorithm. In: **Proceedings of ICCAD**. [S.l.: s.n.], 2005. p. 48–55. ISSN 1092-3152.

PAPA, D. A. et al. Rumble: An incremental timing-driven physical-synthesis optimization algorithm. **TCAD**, v. 27, n. 12, p. 2156–2168, Dec 2008. ISSN 0278-0070.

PILLAGE, L. T.; ROHRER, R. A. Asymptotic waveform evaluation for timing analysis. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 9, n. 4, p. 352–366, Apr 1990. ISSN 0278-0070.

POPOVYCH, S. et al. Density-aware detailed placement with instant legalization. In: **2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)**. [S.l.: s.n.], 2014. p. 1–6. ISSN 0738-100X.

PUGET, J. C. et al. Jezz: An effective legalization algorithm for minimum displacement. In: **Proceedings of SBCCI**. [S.l.: s.n.], 2015. p. 19:1–19:5.

QUINN JR., N. R. The placement problem as viewed from the physics of classical mechanics. In: **Proceedings of DAC**. Piscataway, NJ, USA: IEEE Press, 1975. (DAC '75), p. 173–178.

RABAEY, J.; CHANDRAKASAN, A.; NIKOLIC, B. **Digital integrated circuits: a design perspective**. [S.l.]: Pearson Education, 2003. (Prentice Hall electronics and VLSI series).

REIMANN, T. J. **Cell selection to minimize power in high-performance industrial microprocessor designs**. Thesis (PhD) — Universidade Federal do Rio Grande do Sul, 08 2016.

REN, H.; PAN, D. Z.; KUNG, D. S. Sensitivity guided net weighting for placement-driven synthesis. **IEEE Trans. on CAD**, v. 24, n. 5, p. 711–721, May 2005.

ROY, J. A.; MARKOV, I. L. Eco-system: Embracing the change in placement. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 26, n. 12, p. 2173–2185, Dec 2007. ISSN 0278-0070.

ROY, J. A.; MARKOV, I. L. Eco-system: Embracing the change in placement. In: **2007 Asia and South Pacific Design Automation Conference**. [S.l.: s.n.], 2007. p. 147–152. ISSN 2153-6961.

SAXENA, P.; SHELAR, R.; SAPATNEKAR, S. **Routing Congestion in VLSI Circuits: Estimation and Optimization**. [S.l.]: Springer US, 2007. (Integrated Circuits and Systems). ISBN 9780387485508.

SIPSER, M. **Introduction to the Theory of Computation**. 1st. ed. [S.l.]: International Thomson Publishing, 1996. ISBN 053494728X.

SPINDLER, P.; JOHANNES, F. M. Fast and accurate routing demand estimation for efficient routability-driven placement. In: **2007 Design, Automation Test in Europe Conference Exhibition**. [S.l.: s.n.], 2007. p. 1–6. ISSN 1530-1591.

SPINDLER, P.; SCHLICHTMANN, U.; JOHANNES, F. M. Abacus: Fast legalization of standard cell circuits with minimal movement. In: **Proceedings of the 2008 International Symposium on Physical Design**. New York, NY, USA: ACM, 2008. (ISPD '08), p. 47–53. ISBN 978-1-60558-048-7.

SPINDLER, P.; SCHLICHTMANN, U.; JOHANNES, F. M. Kraftwerk2: A fast force-directed quadratic placement approach using an accurate net model. **IEEE Trans. on CAD**, v. 27, n. 8, p. 1398–1411, Aug 2008.

SUN, W. J.; SECHEN, C. Efficient and effective placement for very large circuits. In: **Proceedings of ICCAD**. [S.l.: s.n.], 1993. p. 170–177.

SWARTZ, W.; SECHEN, C. Timing driven placement for large standard cell circuits. In: **Proceedings of DAC**. New York, NY, USA: ACM, 1995. p. 211–215. ISBN 0-89791-725-1.

TABRIZI, A. F. et al. A detailed routing-aware detailed placement technique. In: **Proceedings of ISVLSI**. [S.l.: s.n.], 2015. p. 38–43. ISSN 2159-3469.

TAU - ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems. 2018. Available from Internet: <www.tauworkshop.com>.

TSAY, R.-S.; KOEHL, J. An analytic net weighting approach for performance optimization in circuit placement. In: **Proceedings of DAC**. [S.l.: s.n.], 1991. p. 620–625.

VISWANATHAN, N.; CHU, C. C. N. Fastplace: efficient analytical placement using cell shifting, iterative local refinement,and a hybrid net model. **IEEE Transactions on CAD of Integrated Circuits and Systems**, v. 24, n. 5, p. 722–733, May 2005.

VISWANATHAN, N. et al. RQL: Global placement via relaxed quadratic spreading and linearization. In: **Proceedings of DAC**. [S.l.: s.n.], 2007. p. 453–458.

VISWANATHAN, N. et al. ITOP: Integrating timing optimization within placement. In: **Proceedings of ISPD**. [S.l.: s.n.], 2010. p. 83–90.

VYGEN, J. Algorithm for large-scale flat placement. In: **Proceedings of the 34th Design Automation Conference**. [S.l.: s.n.], 1997. p. 746–751. ISSN 0738-100X.

WANG, C. H. et al. An effective legalization algorithm for mixed-cell-height standard cells. In: **2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)**. [S.l.: s.n.], 2017. p. 450–455.

WANG, M.; YANG, X.; SARRAFZADEH, M. Dragon2000: standard-cell placement tool for large industry circuits. In: **IEEE/ACM International Conference on Computer Aided Design. ICCAD - 2000. IEEE/ACM Digest of Technical Papers (Cat. No.00CH37140)**. [S.l.: s.n.], 2000. p. 260–263. ISSN 1092-3152.

WANG, Q. B.; LILLIS, J.; SANYAL, S. An lp-based methodology for improved timing-driven placement. In: **Proceedings of ASP-DAC**. [S.l.: s.n.], 2005. v. 2, p. 1139–1143 Vol. 2. ISSN 2153-6961.

WARD, S. I. et al. Structure-aware placement techniques for designs with datapaths. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 32, n. 2, p. 228–241, Feb 2013. ISSN 0278-0070.

WEI, Y. et al. Glare: Global and local wiring aware routability evaluation. In: **DAC Design Automation Conference 2012**. [S.l.: s.n.], 2012. p. 768–773. ISSN 0738-100X.

WEST, D. B. **Introduction to Graph Theory**. [S.l.]: Prentice Hall, 2001. (Featured Titles for Graph Theory Series). ISBN 9780130144003.

WU, G.; CHU, C. Detailed placement algorithm for vlsi design with double-row height standard cells. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 35, n. 9, p. 1569–1573, Sept 2016. ISSN 0278-0070.

WU, G.; CHU, C. Two approaches for timing-driven placement by lagrangian relaxation. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, PP, n. 99, p. 1–1, 2017. ISSN 0278-0070.

ZHANG, Y.; CHU, C. Crop: Fast and effective congestion refinement of placement. In: **Proceedings of the 2009 International Conference on Computer-Aided Design**. New York, NY, USA: ACM, 2009. (ICCAD '09), p. 344–350. ISBN 978-1-60558-800-1.

ZHANG, Y.; CHU, C. Fast and effective placement refinement for routability. **IEEE Trans. on VLSI Systems**, v. 21, n. 9, p. 1751–1756, Sept 2013. ISSN 1063-8210.

ZHAO, W.; CAO, Y. New generation of predictive technology model for sub-45nm design exploration. In: **7th International Symposium on Quality Electronic Design (ISQED'06)**. [S.l.: s.n.], 2006. p. 6 pp.–590. ISSN 1948-3287.

ZHOU, Q.; HU, J.; ZHOU, Q. An effective iterative density aware detailed placement algorithm. In: **Proceedings of ISCAS**. [S.l.: s.n.], 2014. p. 1444–1447. ISSN 0271-4302.

APPENDIX A — EXTRA EXPERIMENTAL RESULTS OF THE PROPOSED NETWORK FLOW-BASED LEGALIZATION ALGORITHM

A.1 Global Placement Results

In this Section, complementary results of the global placement solutions are presented. The number of bins and the distribution of bins by bin types are given in this Section.

A.1.1 RePlace Solution from the 2006 ISPD Benchmarks

In Table A.1, the distribution of bins is presented. In column 1, circuit acronyms are given. In columns 2 to 6, the number of valid bins with the highest 1%, 2%, 5%, 10%, and 20% area density violation are presented, respectively. The total number of overfilled (OF) bins and free of area density violation (Free OF) are given in columns 7 and 8, respectively. In columns 9 and 10, the number of NPA and invalid bins are presented, respectively. The total number of bins is presented in Column 11.

Table A.1: Distribution of bins with ABU violation (ABU 1% to 20%), OF bins, bins free of OF, NPA, invalid and total bins.

Circuit	Number of Bins									
	Area Density Violation					OF	Free OF	NPA	Invalid	Total
	1%	2%	5%	10%	20%					
AD1i	63	126	317	634	1,268	2,780	3,562	983	3,459	9,801
AD2i	76	153	383	767	1,535	2,310	5,367	1,887	9,353	17,030
AD3i	221	443	1,109	2,219	4,439	4,955	17,241	4,948	24,244	46,440
AD4i	288	577	1,443	2,887	5,001	5,001	23,875	3,611	17,564	46,440
BB1i	85	170	426	852	1,705	2,816	5,710	368	1,275	9,801
BB2i	215	431	1,078	2,157	4,314	5,439	16,135	2,112	8,702	30,276
BB3i	620	1,241	3,103	6,206	12,413	19,095	42,970	4,891	27,635	89,700
AD1	63	126	317	634	1,268	2,853	3,489	1,035	3,459	9,801
AD2	76	153	383	767	1,535	2,466	5,211	1,950	9,353	17,030
AD3	221	443	1,109	2,219	4,439	5,189	17,007	4,611	24,244	46,440
AD4	288	577	1,443	2,887	5,294	5,294	23,582	3,557	17,564	46,440
AD5	234	468	1,170	2,340	4,680	10,923	12,478	5,947	23,039	46,440
BB1	85	170	426	852	1,705	2,915	5,611	361	1,275	9,801
BB2	215	431	1,078	2,157	4,314	5,746	15,828	2,122	8,702	30,276
BB4	620	1,241	3,103	6,206	12,413	21,626	40,439	6,396	27,635	89,700
NB5	338	676	1,691	3,382	6,764	16,596	17,225	6,818	22,111	55,932
NB6	510	1,021	2,552	5,105	10,210	11,011	40,040	4,062	15,255	66,306
NB7	722	1,445	3,612	7,225	14,450	20,041	52,210	12,212	58,431	130,682
Avg.	274	550	1,375	2,750	5,430	8,170	19,332	3,771	16,850	44,352

On average, only 274 bins (3.3%) of the overfilled bins (OF) have 11% of the area density utilization violation. On average, 30% and 18% of bins have area density violation compared to the number of valid bins and the total number of bins, respectively. Cells are placed inside 8.5% of invalid bins, on average. On average, bins free of OF are 43.6% of the total number of bins.

A.1.2 FastPlace Solution from the 2006 ISPD Benchmarks

In Table A.2, the distribution of ABU bins is presented. Bins are classified into ABU bins with the highest violation in area density utilization of 1%, 2%, 5%, 10% and 20%, overflowed bins (OF), free of area density violation (Free OF), NPA bins, invalid bins, and total bins. In column 1, circuit acronyms are given. In columns 2 to 6, the number of bins with ABU 1%, ABU 2%, ABU 5%, ABU 10%, and ABU 20% of overfilled bins are presented, respectively. The total number of bins with the overfilled area (OF) and free of the overfilled area (free OF) are given in columns 7 and 8, respectively. In columns 9 and 10, the number of NPA bins and invalid bins are presented, respectively. The total number of bins is presented in column 11.

On average, only 279 of the valid bins have ABU 1% area density violation. On average, 13% of the total bins have an area density violation. NPA bins are 5% of the total ABU bins, on average. On average, 43% of the total bins are free of the ABU violation.

A.1.3 Eh?Placer Solution from the 2006 ISPD Benchmarks

In Table A.3, distribution of ABU bins based on area density violation, invalid ABU bins and total ABU bins are presented. In column 1, acronyms of benchmarks are given. In columns 2 to 6, the number of bins with the highest ABU 1%, ABU 2%, ABU 5%, ABU 10%, and ABU 20% are presented, respectively. The total number of bins with area density violation (OF) and free of area density violation (Free OF) are given in columns 7 and 8, respectively. In columns 9 and 10, the number of NPA bins and invalid bins are presented, respectively. The total number of ABU bins is presented in column 11.

In the Eh?Placer placement solutions, 10% of valid bins have area density violation, on average. On average, NPA bins are 4.3% of the total bins. The total of overfilled

Table A.2: The number of bins with area density violations, free of overfill and invalid ABU. The 2006 ISPD contest benchmarks are placed with the FastPlace global placement algorithm.

Circuit	Number of Bins									
	Area Density Violation					OF	Free OF	NPA	Invalid	Total
	1%	2%	5%	10%	20%					
AD1i	63	126	317	634	1,268	1,859	4,483	669	3,459	9,801
AD2i	76	153	383	767	1,172	1,172	6,505	1,134	9,353	17,030
AD3i	221	443	1,109	2,219	3,088	3,088	19,108	3,699	24,244	46,440
AD4i	288	577	1,443	2,212	2,212	2,212	26,664	1,490	17,564	46,440
BB1i	85	170	426	852	1,213	1,213	7,313	206	1,275	9,801
BB2i	215	431	1,078	2,157	2,609	2,609	18,965	890	8,702	30,276
BB4i	620	1,241	3,103	6,206	8,659	8,659	53,406	1,697	27,635	89,700
AD1	63	126	317	634	1,268	2,775	3,567	518	3,459	9,801
AD2	76	153	383	767	1,535	2,334	5,343	1,866	9,353	17,030
AD3	221	443	1,109	2,219	4,439	5,521	16,675	4,826	24,244	46,440
AD4	288	577	1,443	2,887	4,947	4,947	23,929	2,700	17,564	46,440
AD5	234	468	1,170	2,340	4,680	10,588	12,813	9,749	23,039	46,440
BB1	85	170	426	852	1,705	2,860	5,666	311	1,275	9,801
BB2	215	431	1,078	2,157	4,314	5,992	15,582	1,366	8,702	30,276
BB4	620	1,241	3,103	6,206	12,413	20,637	41,428	4,133	27,635	89,700
NB3	426	852	1,695	1,695	1,695	1,695	40,954	206	124,751	167,400
NB4	207	415	1,038	2,077	4,155	8,322	12,456	2,778	7,952	28,730
NB5	338	676	1,691	3,382	6,764	16,946	16,875	5,995	22,111	55,932
NB6	510	1,021	2,552	5,105	9,432	9,432	41,619	2,036	15,255	66,306
NB7	722	1,445	3,612	7,225	14,157	14,157	58,094	4,615	58,431	130,682
Avg.	279	558	1,374	2,630	4,586	6,351	21,572	2,544	21,800	49,723

Table A.3: The total number of bins are presented. These bins are classified into area density violation, free of overfill area and invalid types. In the 2006 ISPD contest benchmarks, Newblue circuits from 3 to 7 have been placed with the Eh?Placer global placement algorithm.

Circuit	Number of Bins									
	Area Density Violation					OF	Free OF	NPA	Invalid	Total
	1%	2%	5%	10%	20%					
NB3	342	685	1,317	1,317	1,317	1,317	32,956	1,828	99,647	133,920
NB4	249	499	1,249	2,321	2,321	2,321	22,664	2,801	9,355	34,340
NB5	478	957	2,394	4,789	6,032	6,032	41,858	5,112	36,008	83,898
NB6	607	1,214	3,035	5,375	5,375	5,375	55,344	2,955	18,745	79,464
NB7	1,038	2,077	5,193	10,387	11,031	11,031	92,846	9,898	91,965	195,842
Avg.	543	1,086	2,638	4,838	5,215	5,215	49,134	4,519	51,144	105,493

bins is equal to the number of bins which have ABU 20% area density violation. Therefore, area density violation is concentrated in a few bins.

A.1.4 Eh?Placer Solution from the 2015 ICCAD Benchmarks

In Table A.4, distribution of bins based on area density violation, invalid bins and total bins are presented. In column 1, circuit acronyms are given. In columns 2 to 6, the number of bins that have ABU 1%, ABU 2%, ABU 5%, ABU 10%, and ABU 20% area density violation are presented, respectively. The total number of bins with area density violation (OF) is given in column 7. In column 8, the total number of bins free of area density violation (Free OF) are shown. In columns 9 and 10, the number of NPA bins and invalid bins are presented, respectively. The total number of bins is presented in column 11. In column 8, the total number of bins free of area density utilization is shown. In column 9, the total number of invalid bins to compute ABU are presented. A bin is valid to compute ABU only if it has more than 20% of its area available to place movable cells.

Table A.4: The total number of bins are presented. These bins are classified into area density violation, free of overfill area and invalid types. The 2015 ICCAD contest circuit benchmarks are placed with the Eh?Placer global placement algorithm.

Circuit	Number of Bins									
	Area Density Violation					OF	Free OF	NPA	Invalid	Total
	1%	2%	5%	10%	20%					
SB1	586	1,172	1,267	1,267	1,267	1,267	57,379	3,466	54,574	113,220
SB3	636	1,272	3,180	6,361	10,218	10,218	53,396	3,152	65,741	129,355
SB4	444	888	1,420	1,420	1,420	1,420	43,024	3,511	30,996	75,440
SB5	643	990	990	990	990	990	63,313	5,990	107,388	171,691
SB7	856	1,420	1,420	1,420	1,420	1,420	84,266	4,865	52,650	138,336
SB10	255	255	255	255	255	255	97,340	4,530	107,921	205,516
SB16	443	886	925	925	925	925	43,401	2,231	28,707	73,033
SB18	395	458	458	458	458	458	39,095	1,535	20,147	59,700
Avg.	532	918	1,239	1,637	2,119	2,119	60,152	3,660	58,516	120,786

A.2 Legalization Experimental Results

A.2.1 Legalization Results in the RePlace placement solutions from the 2006 ISPD contest benchmarks

In Tables A.5 and A.6, distribution of the total number and percentage of cells by row height displacement are presented, respectively. Cell displacements are ranked from 0 (no displacement) to 5 or higher row heights. In column 1, row height displacements are given. In columns 2 to 19 of Tables A.5 and A.6, the total number and percentage of cell displacements in row heights are presented. Circuits, which have the high-density utilization, have a lower number of legalized cells up to one-row height displacement. On average, inflated circuits (Adaptec 1 to 4 and Bigblue 1, 2, and 3) have twice more cells legalized with the cell displacement of two or more row heights. The maximum area density utilization which impacts ABU (Table 7.3) has a strong correlation with cell displacement (Table A.6). Circuits with high ABU have higher cell displacement.

A.2.2 Legalization Results in the FastPlace placement solutions from the 2006 ISPD contest benchmarks

In Tables A.7 and A.8, the total number of cells and percentage of cells by row height displacement are presented, respectively. In column 1, row height displacements are given. Cell displacements are ranked from 0 (no displacement) to 5 or higher row heights. In this column, left and right numbers indicate the minimum and maximum row height displacement for each range, respectively. In columns 2 to 19, the total number of cells by row height displacement (Table A.7) and percentage of cells by row height displacement (Table A.8) are presented. More cells are legalized with high cell displacement in circuits that have high-density utilization of fixed cells and high maximum NPA. In these circuits, cell displacement is evenly distributed up to two-row heights.

Table A.5: Distribution of total cells by row height displacement for circuits from the 2006 ISPD benchmarks.

Disp.	The number of Cells by Row Height Displacement ($\times 10^3$)																	
	AD1i	AD2i	AD3i	AD4i	BB1i	BB2i	BB4i	AD1	AD2	AD3	AD4	AD5	BB1	BB2	BB4	NB5	NB6	NB7
No	3.2	3.2	6.4	7.8	4.7	8.3	36.5	8.6	8.3	17.0	18.9	34.5	11.0	20.6	82.4	44.2	25.6	55.3
0-1	123.9	137.3	256.6	287.9	170.7	317.0	1446.0	171.6	197.6	362.4	398.8	706.9	228.9	439.6	1889.1	993.1	826.7	1814.5
1-2	69.9	85.4	147.6	160.3	87.2	167.4	604.1	29.0	42.5	63.5	69.7	92.0	36.2	68.1	185.9	176.1	336.9	544.8
2-3	11.2	19.2	26.6	28.3	12.1	28.9	65.2	0.9	2.8	2.5	2.8	2.4	1.0	3.2	3.7	5.8	45.1	43.8
3-4	1.4	3.8	4.8	4.3	1.3	5.8	8.1	0.1	0.5	0.6	0.5	0.5	0.1	0.5	0.7	1.1	6.5	5.4
4-5	0.4	1.1	1.8	1.3	0.4	2.2	2.5	0.1	0.3	0.4	0.4	0.5	0.1	0.3	0.6	0.8	1.9	2.2
5+	0.8	4.5	7.1	4.8	1.2	5.1	6.8	0.6	2.6	4.5	3.5	5.6	0.3	2.4	6.7	7.0	5.4	15.4

Table A.6: Distribution of cells by row height displacement for circuits from the 2006 ISPD benchmarks.

Disp.	Cell Distribution by Disp. (%)																	
	AD1i	AD2i	AD3i	AD4i	BB1i	BB2i	BB4i	AD1	AD2	AD3	AD4	AD5	BB1	BB2	BB4	NB5	NB6	NB7
No	1.5	1.3	1.4	1.6	1.7	1.6	1.7	4.1	3.3	3.8	3.8	4.1	4.0	3.9	3.8	3.6	2.0	2.2
0-1	58.8	54.0	56.9	58.2	61.5	59.3	66.7	81.3	77.6	80.4	80.6	83.9	82.4	82.2	87.1	80.9	66.2	73.1
1-2	33.1	33.6	32.7	32.4	31.4	31.3	27.9	13.7	16.7	14.1	14.1	10.9	13.0	12.7	8.6	14.3	27.0	22.0
2-3	5.3	7.5	5.9	5.7	4.4	5.4	3.0	0.4	1.1	0.6	0.6	0.3	0.4	0.6	0.2	0.5	3.6	1.8
3-4	0.7	1.5	1.1	0.9	0.5	1.1	0.4	0.1	0.2	0.1	0.1	0.1	0.0	0.1	0.0	0.1	0.5	0.2
4-5	0.2	0.4	0.4	0.3	0.1	0.4	0.1	0.0	0.1	0.1	0.1	0.1	0.0	0.1	0.0	0.1	0.2	0.1
5+	0.4	1.7	1.6	1.0	0.4	1.0	0.3	0.3	1.0	1.0	0.7	0.7	0.1	0.4	0.3	0.6	0.4	0.6

Table A.7: Distribution of total cells by row height displacement

Disp.	Total of Cells by Row Height Displacement of Each Circuit $(\times 10^3)$																			
	AD1i	AD2i	AD3i	AD4i	BB1i	BB2i	BB4i	AD1	AD2	AD3	AD4	AD5	BB1	BB2	BB4	NB3	NB4	NB5	NB6	NB7
No	12.9	28.6	43.0	79.2	43.5	73.3	304.4	46.3	51.4	71.1	121.6	107.3	80.0	118.8	462.1	142.4	148.9	174.4	204.7	434.8
0-1	40.6	70.8	77.6	128.0	78.3	137.5	693.0	68.0	90.3	119.3	168.7	201.1	101.9	188.7	847.3	134.9	215.1	365.5	417.1	930.5
1-2	57.8	77.1	101.5	158.4	91.7	174.5	693.1	62.9	66.4	108.3	131.3	159.7	69.9	156.8	532.2	150.6	165.5	258.1	356.1	692.1
2-3	34.0	32.9	50.1	62.0	33.3	72.2	248.5	19.0	19.5	39.1	32.3	56.4	15.0	41.8	134.4	36.8	46.1	84.7	115.2	211.3
3-4	19.3	14.7	26.7	26.3	12.6	31.0	95.4	6.7	6.9	18.6	10.8	27.5	4.3	13.1	46.8	10.2	16.8	39.2	44.8	76.3
4-5	11.1	7.6	16.7	13.2	5.7	15.2	43.9	2.9	3.5	11.1	5.0	17.5	1.8	5.3	23.7	3.6	8.3	24.8	22.9	35.3
5+	35.2	22.8	135.3	27.6	12.5	31.1	90.9	5.1	16.6	83.5	25.0	273.1	4.7	10.3	122.7	4.5	42.2	281.6	87.4	101.0

Table A.8: Distribution of cells by row height displacement for circuits from the 2006 ISPD benchmarks.

Disp.	Cell Distribution by displacement in terms of row height (%)																			
	AD1i	AD2i	AD3i	AD4i	BB1i	BB2i	BB4i	AD1	AD2	AD3	AD4	AD5	BB1	BB2	BB4	NB3	NB4	NB5	NB6	NB7
No	6.1	11.2	9.5	16.0	15.7	13.7	14.0	22.0	20.2	15.8	24.6	12.7	28.8	22.2	21.3	29.5	23.2	14.2	16.4	17.5
0-1	19.2	27.8	17.2	25.9	28.2	25.7	31.9	32.2	35.5	26.5	34.1	23.9	36.7	35.3	39.1	27.9	33.5	29.8	33.4	37.5
1-2	27.4	30.3	22.5	32.0	33.0	32.6	32.0	29.8	26.1	24.0	26.5	19.0	25.2	29.3	24.5	31.2	25.7	21.0	28.5	27.9
2-3	16.1	12.9	11.1	12.5	12.0	13.5	11.5	9.0	7.7	8.7	6.5	6.7	5.4	7.8	6.2	7.6	7.2	6.9	9.2	8.5
3-4	9.1	5.8	5.9	5.3	4.6	5.8	4.4	3.2	2.7	4.1	2.2	3.3	1.6	2.4	2.2	2.1	2.6	3.2	3.6	3.1
4-5	5.3	3.0	3.7	2.7	2.0	2.8	2.0	1.4	1.4	2.5	1.0	2.1	0.6	1.0	1.1	0.7	1.3	2.0	1.8	1.4
5+	16.7	9.0	30.0	5.6	4.5	5.8	4.2	2.4	6.5	18.5	5.1	32.4	1.7	1.9	5.7	0.9	6.6	22.9	7.0	4.1

A.2.3 Legalization Results in the Eh?Placer placement solutions from the 2006 ISPD contest benchmarks

In Tables A.9 and A.10, the total number and percentage of cells by row height displacement are presented, respectively. Cell displacements are ranked from 0 (no displacement) to 5 or higher row heights. In column 1, row height displacements are given. In columns 2 to 6, the total number and percentage of cell displacements given in row heights are presented.

Table A.9: Distribution of total cells by row height displacement for Newblue 3 to 7 circuits from the 2006 ISPD benchmarks.

Disp.	Cell Displacement Distribution ($\times 10^3$)				
	NB3	NB4	NB5	NB6	NB7
No	8.0	8.4	10.0	12.3	33.7
0-1	226.0	334.8	622.5	647.7	1340.7
1-2	138.2	194.1	377.3	401.7	745.1
2-3	53.8	57.2	114.8	112.3	202.4
3-4	24.9	21.3	42.7	37.3	68.9
4-5	13.9	10.2	20.4	16.4	30.3
5+	18.0	16.7	40.4	20.4	60.2

Table A.10: Distribution of cells by row height displacement for Newblue 3 to 7 circuits from the 2006 ISPD benchmarks.

Disp.	Cell Distribution by Disp. (%)				
	NB3	NB4	NB5	NB6	NB7
No	1.7	1.3	0.8	1.0	1.4
0-1	46.8	52.1	50.7	51.9	54.0
1-2	28.6	30.2	30.7	32.2	30.0
2-3	11.1	8.9	9.3	9.0	8.2
3-4	5.2	3.3	3.5	3.0	2.8
4-5	2.9	1.6	1.7	1.3	1.2
5+	3.7	2.6	3.3	1.6	2.4

A.2.4 Legalization Results in the Eh?Placer placement solutions from the 2015 ICCAD contest benchmarks

In Tables A.11 and A.12, the total number and percentage of cells by row height displacement are presented, respectively. Cell displacements are ranked from 0 (no displacement) to 5 or higher row heights. In column 1, row height displacements are given. In columns 2 to 9, the total number and percentage of cell displacements in terms of row heights are presented. More cells are legalized with high cell displacement in circuits that have high-density utilization of fixed cells and high maximum NPA.

Table A.11: Distribution of total cells by row height displacement for circuits from the 2015 ICCAD benchmarks.

Disp.	Total of Cells by Row Height Disp. $(\times 10^3)$							
	SB1	SB3	SB4	SB5	SB7	SB10	SB16	SB18
No	0.2	0.0	0.1	0.1	0.1	0.1	0.0	0.1
0-1	878.9	463.0	517.5	803.1	1334.7	1418.0	683.6	565.1
1-2	264.6	383.6	196.2	212.5	467.4	379.7	245.8	164.9
2-3	40.5	171.7	45.5	35.7	79.7	56.6	38.9	24.2
3-4	9.9	75.1	14.6	11.6	18.9	10.8	6.8	5.8
4-5	4.2	37.8	6.4	6.1	7.9	3.7	2.2	2.6
5+	7.7	80.0	11.9	15.8	18.0	5.4	4.1	4.7

Table A.12: Percentage of cell distribution by row height displacement for circuits from the 2015 ICCAD benchmarks.

Disp.	Cell Distribution by Disp. (%)							
	SB1	SB3	SB4	SB5	SB7	SB10	SB16	SB18
No	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0-1	72.9	38.2	65.3	74.0	69.3	75.7	69.7	73.6
1-2	21.9	31.7	24.8	19.6	24.3	20.3	25.0	21.5
2-3	3.4	14.2	5.7	3.3	4.1	3.0	4.0	3.2
3-4	0.8	6.2	1.8	1.1	1.0	0.6	0.7	0.8
4-5	0.3	3.1	0.8	0.6	0.4	0.2	0.2	0.3
5+	0.6	6.6	1.5	1.5	0.9	0.3	0.4	0.6

APPENDIX B — ELECTRONIC DESIGN AUTOMATION CONTESTS

EDA contests provide an environment for the academy and industry to collaborate and achieve innovations in challenging and complex design problems (DARAV et al., 2015). Over the past twelve years, a set of EDA contests were promoted by ISPD (ISPD..., 2017), ICCAD (ICCAD..., 2017) and International Workshop on Power And Timing Modeling, Optimization and Simulation (PATMOS)/International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU) (TAU..., 2018) conferences. These contests mainly address physical synthesis subjects, such as placement, routing, CTS, timing closure, and so forth. Usually, leading industrial companies propose the contest topic and organize it.

In each contest, the specification of a relevant EDA problem, circuit benchmarks, and evaluation metric are provided. Usually, each contest edition takes up to 9 months from the subject announcement to the submission of the final tool or solution.

In Appendix B.2, a detailed list of contest subjects promoted by ISPD, ICCAD and PATMOS/TAU is presented. The focus of the ISPD and ICCAD contests is on the physical synthesis of the ASICs. Recently, ISPD promoted two contest editions targeted to FPGAs placement optimization. PATMOS/TAU focuses is on power and timing modeling, optimization and simulation for ASICs.

B.1 Circuit Benchmarks

Benchmarks provide a standard field to measure and compare algorithms development (NAM; CONG, 2007). However, one may focus on tuning the algorithm to provide the best solution instead of focusing on a general purpose solution. Therefore, the proposed algorithm becomes problematic to provide a proper solution for any circuits.

On the other hand, usually, the contest benchmarks are limited to the proposed EDA problem. It is very uncommon to have benchmarks that provide extra data. Therefore, there is a lack of circuits that comprise the entire physical synthesis flow, i.e., benchmarks have only the required files for the target optimization problem.

In the nineties, the best known benchmarks were International Symposium on Circuits and Systems (ISCAS)-85 and ISCAS-89 that are available at (ACM/SIGDA..., 2017) and ISPD-98 (ALPERT; MEHTA; SAPATNEKAR, 2008). In 2004, (CHANG et al., 2004) built Placement Examples with Known Optimal (PEKO) circuits where the optimum placement solutions are know. They have evaluated Dragon (WANG; YANG; SARRAFZADEH, 2000), Capo (CALDWELL; KAHNG; MARKOV, 2000), mPL (CHAN et al., 2000), mPG (CHANG; CONG; PAN, 2002) and QPlace (ENVISIA. . . , 1999) placement algorithms using PEKO. Evaluated placement algorithms had significant variations in the placement solution and an enormous gap in the optimum solution. On the other hand, PEKO benchmarks have only local connections and may not reflect the structures found in real circuits. The results from (CHANG et al., 2004) have highlighted the urgency to have relevant benchmarks and standard metrics to evaluated EDA algorithms. Therefore, several contest editions were made to target specific physical synthesis state-of-the-arts problems. Each edition released its set of circuits.

Usually, circuits have been released using Bookshelf format (BOOKSHELF, 2017). Recently, benchmarks have been released using industrial formats (e.g., Layout Exchange Format (LEF), Design Exchange Format (DEF), Verilog, Standard Parasitic Exchange Format (SPEF), and Liberty). The circuits in the industrial file format are mainly built using NanGate FreePDK45 Generic Open Cell Library (NANGATE. . . , 2017). The cell library is synthetic and built based on 45nm Predictive Technology Model (PTM) (ZHAO; CAO, 2006). A 15nm library cell was recently released (MARTINS et al., 2015) similar to FreePDK45 library.

B.2 List of Contest Subjects

In Tables B.1, B.2, B.3, andB.4, a detailed list of the EDA contests promoted by ISPD, PATMOS/TAU, DAC, and ICCAD conferences are presented, respectively.

DAC promoted only one edition of the EDA contest. In Table B.3 is presented the subject of the contest.

Table B.1: The EDA contests in past years hosted by ISPD

Year	Subject
2005	Placement
2006	Placement
2007	Global Routing
2008	Global Routing
2009	Clock Network Synthesis
2010	High Performance Clock Network Synthesis
2011	Routability-Driven Placement
2012	Discrete Gate Sizing Contest
2013	Discrete Gate Sizing Contest
2014	Detailed Routing-Driven Placement Contest
2015	Blockage-Aware Detailed Routing-Driven Placement Contest
2016	Routability-Driven FPGA Placement Contest
2017	Clock-Aware FPGA Placement
2018	Initial Detailed Routing
2019	Initial Detailed Routing

Table B.2: The EDA contests in past years hosted by TAU and PATMOS

Year	Subject
2011	Timing Analysis Contest (PATMOS)
2013	Variation Aware Timing Analysis Contest
2014	Removing Pessimism during Timing Analysis
2015	Incremental Timing and Common Path Pessimism Removal (CPPR) Analysis
2016	Timing Macro Modeling
2017	Timing Macro Modeling
2018	Efficient generation of timing reports from an STA graph with updated arrival and required times
2019	Timing-driven optimization

Table B.3: The EDA contests in past years hosted by DAC

Year	Subject
2012	Routability-Driven Placement Contest and Benchmark Suite

Table B.4: The EDA contests in past years hosted by ICCAD

Year	Subject
2012	Finding the minimal logic difference for functional ECO Design hierarchy aware routability-driven placement Fuzzy pattern matching for physical verification
2013	Technology Mapping for Macro Blocks contributed Placement Finishing – Detailed Placement and Legalization contributed Mask Optimization
2014	Simultaneous CNF Encoder Optimization with SAT solver Setting Selection Incremental Timing-driven Placement Design for Manufacturability Flow for Advanced Semiconductor Nodes
2015	3D-ICON: 3D Interlayer Cooling Optimized Network Large-Scale Equivalence Checking and Function Correction Incremental Timing-driven Placement
2016	Identical Fault Search NP3: Non-exact Projective NPNP Boolean Matching Pattern Classification for Integrated Circuit Design Space Analysis
2017	Resource-aware Patch Generation Net Open Location Finder with Obstacles Multi-Deck Standard Cell Legalization
2018	Smart EC: Program-Building for Name Mapping Obstacle-Aware On-Track Bus Routing Timing-Aware Fill Insertion
2019	Logic Regression on High Dimensional Boolean Space System-level FPGA Routing with Timing Division Multiplexing Technique LEF/DEF Based Open-Source Global Router