Design for Manufacturability aware Early Global Routing

Bapi Kar

Design for Manufacturability aware Early Global Routing

Thesis submitted in partial fulfillment
for the requirements of the degree

of

Doctor of Philosophy

by

Bapi Kar

Under the guidance of

Prof. Chittaranjan Mandal

and

Prof. Susmita Sur-Kolay

COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR, INDIA

KHARAGPUR - 721 302, INDIA

June 2017

To my daughter Mrittika, wife Srabanti, mother Pronati and a motherly person.

Certificate of Approval

___ /___ /_____

Certified that the thesis entitled **Design for Manufacturability aware Early Global Routing** submitted by **Bapi Kar** to the Indian Institute of Technology, Kharagpur, for the award of the degree Doctor of Philosophy has been accepted by the external examiners and that the student has successfully defended the thesis in the viva-voce examination held today.

(Supervisor) (Co-Supervisor)

(Member of the DSC) (Member of the DSC) (Member of the DSC)

(External Examiner)

(Chairman)

Certificate

This is to certify that the thesis entitled "**Design for Manufacturability aware Early Global Routing**" submitted by **Bapi Kar** to Indian Institute of Technology, Kharagpur, is a record of bona fide research work carried out under our joint supervision and is worth considering for the award of the degree of Doctor of Philosophy of the Institute.

Prof. Chittaranjan Mandal
Indian Institute of Technology, Kharagpur
Kharagpur -721 302, INDIA

Prof. Susmita Sur-Kolay
Indian Statistical Institute, Kolkata
Kolkata -700108, INDIA

Declaration

I do hereby declare that

a. the work contained in this thesis is original and has been done by me under the guidance of my Supervisor(s);
b. the work has not been submitted to any other Institute for any degree or diploma;
c. I have followed the guidelines provided by the Institute in preparing the thesis;
d. I have conformed to ethical norms and guidelines while writing the thesis and;
e. whenever I have used materials (data, models, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis, giving their details in the references, and taking permission from the copyright owners of the sources, whenever necessary.

Bapi Kar

Acknowledgments

I start this section by paying my gratitude to all the teachers and the mothers of this world, without whom we all are non-existent. I am indebted to my supervisors Prof. Susmita Sur-Kolay of Indian Statistical Institute, Kolkata and Prof. Chittaranjan Mandal of Indian Institute of Technology, Kharagpur for their never-ending technical help, mental support, parental guidance, inspiration for life and affection for doing anything and everything related to this work.

In this part of my life, I am indebted to many people both in IIT Kharagpur and ISI Kolkata whose tremendous support and encouragement have helped me sail through this journey. I would specifically like to name with reverence, Prof. Dipankar Sarkar, Prof. Bhargab Bhattacharya, Prof. Parthapratim Chakraborty, Prof. Subhash Nandy, Prof. Nabanita Das, Prof. Bimal Roy, Prof. Sanghamitra Bandopadhyaya, Prof. Debashis Samanta, Prof. Siddhartha Sen, Prof. Santanu Chattapadhayay, Prof. Niloy Ganguly, Prof. Karabi Biswas and other respectable teachers of both the institutes, as a part of expressing my heartfelt gratitude towards them. It is also impossible to forget the uncountable assistance from the office staffs of both these institutes.

My sincere regard goes to Sridhar Rangarajan, Vamsimohan Kasturi, Subhrojit Bhattacharya and few others at IBM India for giving me the opportunity to engage with them and work in their electronic design automation (EDA) flow. My special thanks to my Mentor Dr. Cliff Sze of IBM at Texas Austin Lab for his guidance, encouragement and inspiration to work in the field of physical design automation during my internship at IBM. This list also includes a group of people from Mentor Graphics Corporation, namely Nilanjan Mukherjee, Sarvesh Bhardwaj, Atul Dogra and his team, for their generous help towards a part of this thesis, by providing the access to Mentor's EDA tool setup and technical help. I am also grateful to Kaushik Dutta, Goutam Bhaumik and their team at Mentor Graphics for their generous help.

Bapi Kar

Abstract

In this thesis, we present the first of its kind early global routing framework after floorplanning of a design is done as per the existing physical design (PD) flow. The intent of this work is to assess early routability of a design, earlier than the existing practices in the present PD flow. The proposed routing framework begins with the identification of a set of monotone staircase routing regions obtained by recursive floorplan bipartitioning, based on a simple floorplan topology graph. For this purpose, a new floorplan bipartitioning framework based on different graph search techniques on the floorplan graph is proposed for faster completion, as compared to the existing maxflow based methods that incur higher runtime overhead. During the proposed early global routing, we aim to address an early version of unconstrained via minimization (UVM) problem, by an improved bipartitioning framework for identifying of a set of minimal bend monotone staircase routing regions, using a greedy and a randomized search technique.

The corresponding routing regions are used to construct a new routing graph model which allows the nets to be strictly routed through these routing regions in a number of routing (metal) layers. Notably, this routing model is fundamentally different from the grid graph model used in the existing global routing methods. Subsequently, we extend this early routing model for undertaking over-the-block early global routing at floorplan level. This hybrid model employs the monotone staircases for routing in lower routing layers, while a suitable floorplan based adaptation of the existing grid graph model is used for upper layers. Unlike the existing grid graph model used in post-placement global routing, both the proposed routing models can potentially address the pin accessibility problem of the nets at floorplan block level by suitable edge definition. Beside early routability assessment, both the models ensure that the congestion in any routing region is restricted to 100% and also adhere to the proposed UVM based early routing topology generation and layer assignment of the nets for minimal via routing. The corresponding routing solutions can be helpful in guiding the subsequent post-placement global/detailed routing as well as performing detailed placement of standard cells while obeying the given floorplan topology. In

this regard, a case study on different floorplan instances of an industrial design with the help of a well known industrial physical design (PD) tool has been also conducted.

Finally, the proposed routing framework is explored for design for manufacturability (DFM) aware early routability assessment, by considering an early abstraction model of edge placement errors (EPE) due to the limitation of planar fabrication processes. In this part, we also focus on uniform wire distribution intended for minimizing the surface irregularities due to non-uniform metal density across the layout and different hardness factors of the metal and the dielectric materials during chemical mechanical polishing (CMP) process.

Keywords- Electronic design automation, physical design flow, floorplanning, global routing, routing region definition, monotone staircase regions, pin access, congestion, unconstrained via minimization, chemical mechanical polishing, design for manufacturability, edge placement error.

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

$2D$: Two Dimensional

$3D$: Three Dimensional

β: Trade off parameter for obtaining minimal bends monotone staircase routing regions

ϵ: Weight bound parameter for area (number) balanced bipartitioning

γ: Trade off parameter for maximizing area (number) balance in floorplan bipartitioning

λ: Wavelength of ArF laser used in optical lithograghy

ArF: Argon Fluoride Laser

B: Set of blocks in a floorplan

b_i: A block in a floorplan

baltype: Area/Number balance type in floorplan bipartitioning

d: Critical dimension of a geometric feature in the design layout

E_G: Set of edges in a graph G

e_j: An edge in a graph G

F: An input floorplan instance

M_i: A permissible metal (routing) layer

N: Set of nets in a floorplan

n_j: A net in a floorplan

stype: MIS/MDS staircase type

t_k: A pin (terminal) in a net

V_G: Set of vertices in a graph G

v_i: A vertex in a graph G

ASIC: Application Specific Integrated Circuit

BAG: Block Adjacency Graph

BC: Best Case

BFS: Breadth First Search

CMP: Chemical Mechanical Polishing

CVM: Constrained Via Minimization

DEF: Design Exchange Format

DFM: Design for Manufacturability

DFS: Depth First Search

DoF: Dept of Focus

DPL: Double Patterning Lithography

DRC: Design Rule Check

EBM: Electron Beam Lithography

ECO: Engineering Change Order

EDA: Electronic Design Automation

EGR: Early Global Routing

EPE: Edge Placement Error

EUV: Extreme Ultraviolet Laser

GCell: A global routing bin/cell/tile in grid graph model

GSRG: Global Staircase Routing Graph

hGSRG: Hybrid Global Staircase Routing Graph

HPWL: Half Perimeter Wirelength

IC: Integrated Circuit

LEF: Library Exchange Format

LVS: Layout versus Schematic Check

MDS: Monotone Decreasing Staircase

MIS: Monotone Increasing Staircase

MSC: Monotone Staircase Cut

MST: Minimum Spanning Tree

NB: No Bend

OAI: Off-Axis Illumination

OPC: Optical Proximity Correction

OPE: Optical Proximity Error

ORC: Optical Rule Check

PD: Physical Design

PSM: Phase Shift Masks

RET: Resolution Enhancement Techniques

RLM: Reserved Layer Model

RMST: Rectilinear Minimal Spanning Tree

RR: Ripup and Reroute

RSMT: Rectilinear Steiner Minimal Tree

SMST: Staircase Minimal Steiner Topology

SoC: System on Chip

TNS: Total Negative Slack

TPL: Tripple Patterning Lithography

TSV: Through Silicon Via

ULM: Unreserved Layer Model

UVM: Unconstrained Via Minimization

VDSM: Very Deep Sub-Micron

VLSI: Very Large Scale Integration

WC: Worst Case

WD: Wire Distribution

WNS: Worst Negative Slack

Chapter 1

Introduction

Integrated Circuit (IC) design flow comprises of many constituent steps, as illustrated in Figure 1.1. Initial stages in this flow mainly focus on functional realization of a design and defining its performance oriented objectives like functional specification, architecture design, followed by functional/structural logic design and verification. On successful completion with a prescribed degree of satisfaction, physical implementation of the design is initiated for layout generation for a targeted fabrication process node. The physical design (PD) flow [17] consists of major optimization stages such as floorplanning, placement, and global/detailed routing. After successful sign-off at the physical verification stage, the PD flow produces a layout of the design comprising of geometrical shapes only, which is sent to the fabrication process for mask generation for the layout. These masks are used to transfer these geometric shapes on to the silicon wafer using the existing optical lithography system. Subsequently, the wafers consisting multiple die are tested using industrial testers so that the fault-free die can be identified for packaging. Better implementation during the PD flow and proper fabrication of the design layout lead to more working die, thereby increasing *manufacturing yield*.

IC fabrication process technology continues to evolve due to increasing demand of numerous functionality in a design, such as System-On-Chip (SoC) and Network-On-Chip (NoC) designs, in order to integrate more number of design components in a specified die area. Despite this miniaturization drive, the optical lithography system continues to use the same ArF laser system having $193nm$ wavelength for smaller (very deep submicron) technology nodes such as $65nm$ and below (see Figure 1.2). This is commonly known as *process design gap*. This makes the desired printability

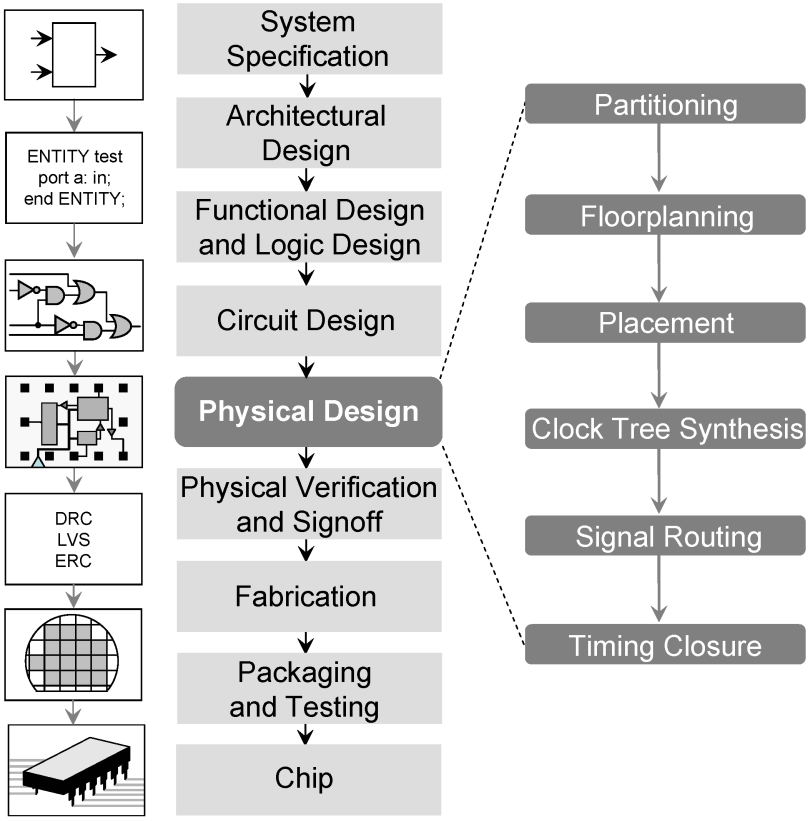


Figure 1.1: A complete Integrated Circuit (IC) Design Flow [1]

of the constituent sub-wavelength features difficult, causing the desired geometries of the features to deviate from their desired shape and size. These are collectively known as *design for manufacturability* (DFM) issues and become more prominent as the technology progresses to smaller nodes. The printability of sub-wavelength feature sizes have aggravated the existing challenges in physical design (PD) (see Figure 1.1), putting more burden on physical verification due to increased design and the lithography rules. This increases the corresponding efforts in post-route layout optimization due to these printibility issues by many folds, in addition to increasing design iterations for mitigating these issues during placement and routing optimization.

Later in this chapter, we discuss another important manufacturability issue arising due to chemical mechanical polishing (CMP) during IC fabrication process, causing several structural as well as functional issues. Here, we also consider another important issue, related to the number of vias used for multi-layer interconnections of the

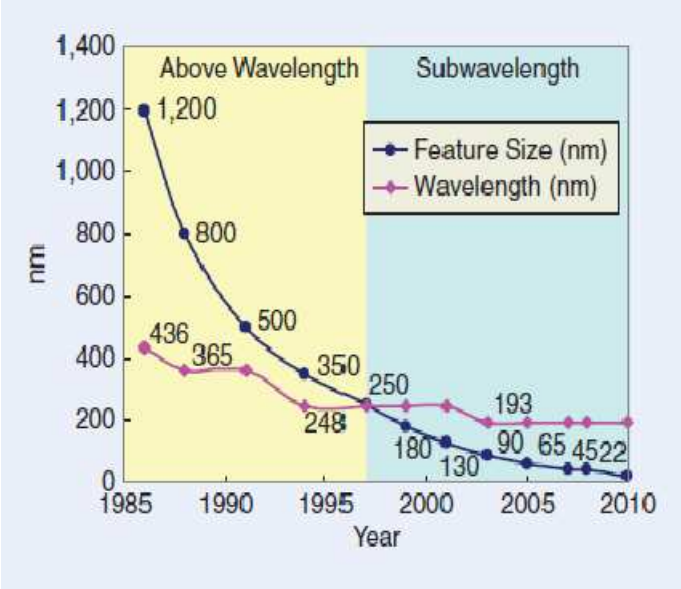


Figure 1.2: Process Design Gap: Evolution of fabrication process technology nodes with respect to optical lithography system [2, 3]

net segments, known as design for reliability issue.

1.1 Design for Manufacturability Issues

A schematic overview of the existing optical lithography system is depicted in Figure 1.3, consisting of an ArF light source of a specified wavelength ($\lambda = 193nm$). The light from this source is directed through the mask of a given layer, for printing the desired geometric patterns on the wafer, using a forward projection lens. Another projection lens is employed after the mask in order to focus the passing rays of light, coming out of the designated openings in the mask, on the photo-resist coating applied on the wafer surface for printing the desired patterns. A geometric feature with a dimension d can be printed with little deformation when λ is significantly smaller than d, while significant deviation from the original shape and size is evident when the feature size d is too small as compared to λ. This is illustrated in Figure 1.4 (a) and happens due to optical diffraction effect at the edges of the respective mask openings.

As per the trend between the optical wavelength and the critical feature dimension presented in the process-design gap diagram in Figure 1.2, technology nodes with higher d values such as those above $180nm$ used a higher λ value than for the nodes

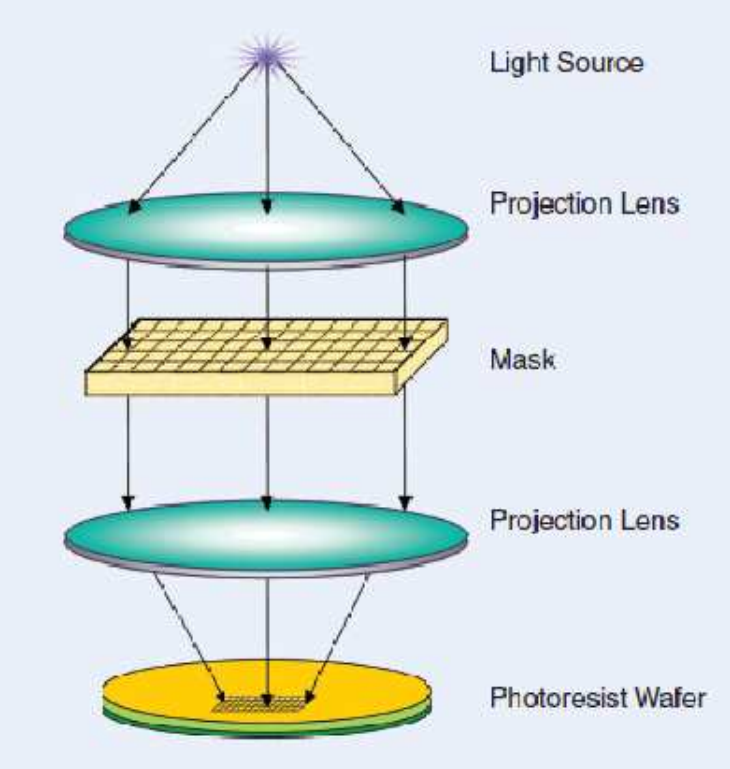


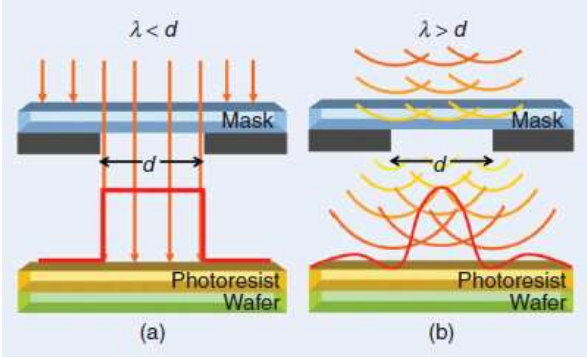
Figure 1.3: A schematic of the existing optical lithography system [2]

in $180nm$ and below. However, further reduction in critical dimension of features in subsequent smaller nodes, such as $65nm$ and below, did not come up with a proportional reduction in wavelength (λ), but remained to be constant at $193nm$ for $90nm$ and below. This trend causes prominent optical diffraction effect in the very deep submicron technology nodes, those below $90nm$ node. As a result, mitigating the said geometric deformations turned out to be more challenging in successful design closure, causing various functional and structural failures in the fabricated die.

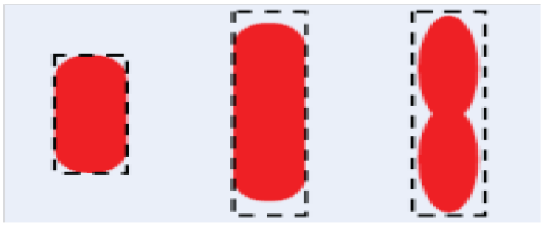
1.1.1 Limitations of Optical Lithography System

The geometric deformations of the features due to the said limitation of the optical lithography system cause severe degradation in electrical properties such as intra/inter layer parasitic capacitance and wire resistance. These errors also induce structural issues in the wires such as open and shorts faults, bridging faults. In order to minimize these errors, several resolution enhancement techniques (RET) are popularly being used such as rule/model based optical proximity correction (OPC) techniques

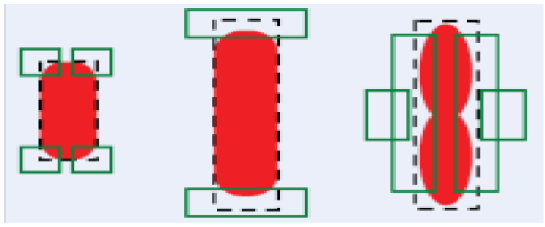
(see Figure 1.4), double/triple patterning based multi-mask synthesis for the same routing layer, simulation based routing blockage generation due to edge placement error (EPE) etc. Post-layout RET approaches aim at substantial, if not complete, removal of these lithographic violations. However, it requires several iterations before an acceptable solution with minimal/no such violations is obtained. During these iterations, many of the violations related to design rule checks (DRC), optical rule checks (ORC), simulation based OPE/EPE checks are identified.



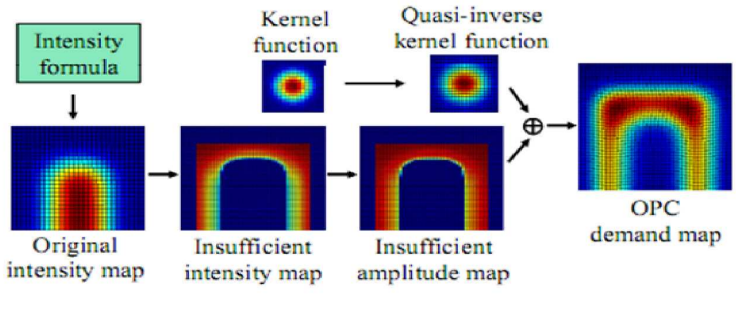
(a) Sub-wavelength diffraction effect



(b) Geometric deformation



(c) Rule-based OPC



(d) Simulation-based OPC

Figure 1.4: Sub-wavelength feature printing and Optical Proximity Errors/Correction (OPE/OPC) [2, 4]

During the iterative optimization process, it is desirable that all such violations are resolved before the layout freezes for fabrication. Practically, no such guarantee

can be made towards a violation free design implementation within a stipulated time, thus impacting both the manufacturing yield and design turnaround time. Moreover, it is very difficult to resolve these issues if discovered much later in the IC design flow. In many occasions, a designer solves some of the issues, if not all, manually based on his experience. Therefore, it is recommended and highly desirable to estimate one or more such issues earlier during the physical design flow. Recent research works considering DFM issues like OPC/EPE and CMP were proposed in order to handle them during the detailed routing, mainly by incremental methods such as wire-spreading or rip-up and re-route (RR) techniques. A few such works on similar DFM aware optimization approaches were also suggested during global routing, with intent to reduce the number of iterations and thus design turn around time.

Recent approaches of interleaved design optimization in the PD flow have also gained momentum and are shown to be effective than the earlier straight forward approaches. Efforts have been made for the optimization issues like placeability, routability, and timing in the earlier design stages by suitable modeling. These effects may subsequently be used as a set of constraints for iterative global/detailed routing, placement, physical synthesis and even Engineering Change Order (ECO) in litho-friendly design paradigm. Similarly, in case of lithographic violations, the intent is to consider them as early as possible during different stages of the physical design flow rather than pushing them all to the post-layout optimization stage.

In order to alleviate the limitation of the existing $193nm$ optical lithography system, evolution of other lithographic systems such as Extreme Ultraviolet (EUV) Lithography, Electronic Beam Lithography (EBL) and Immersion Lithography (IL) system are also gaining momentum for enhancing manufacturing yield. These systems are projected to facilitate better litho-friendly design (LFD) coverage over the existing system for very deep submicron (VDSM) technology nodes such as $22nm$ and below. Despite that, they are yet to be placed for full scale commercial operation in order to immediately replace the existing system due to several technical as well as economical reasons. A schematic depiction of Immersion lithography system is presented in Figure 1.5.

1.1.2 Chemical Mechanical Polishing

Beside the DFM issues induced by the existing optical lithography system, chemical mechanical polishing (CMP) step (see Figure 1.6) during the fabrication process

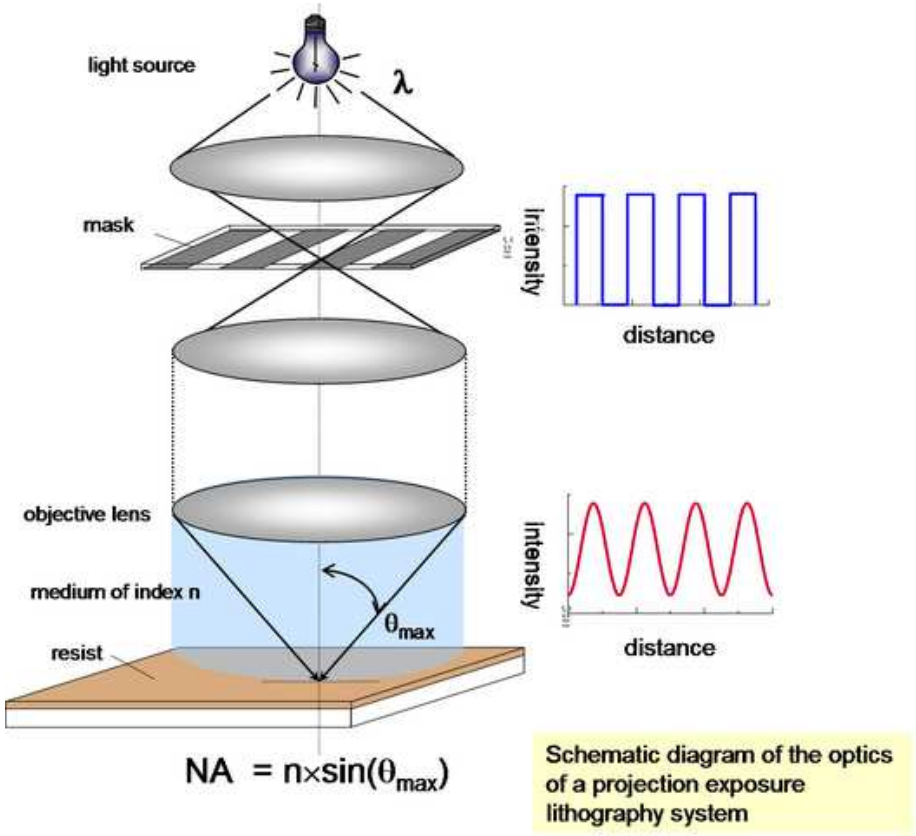


Figure 1.5: Schematic of Immersion Lithography (IL) system (IBM Research) [5]

too greatly impacts the manufacturability of a design. This process causes irregular surface at each routing layer due to (i) nonuniform metal (wire) density across the metal layers, and (ii) the difference in the hardness factors of the dielectric material (silicon oxide) and the metal deposits. Irregular polishing of these routing layers (see Figure 1.7) deeply impacts the timing as well as the power budget in a design resulted from unpredictable inter-layer parasitic capacitance and non-uniform resistance due to irregular metal thickness. This effect is also accompanied by inconsistent metal width due to optical proximity errors (OPE) as highlighted in previous section.

In order to minimize the CMP effects, such as erosion and dishing effects, the traditional practice has been to insert dummy metal fills across the layout in order to maintain a uniform metal density. This is intended to ensure the inter-layer parasitic capacitance values to be more deterministic and ensures more uniformity in wire thickness in a given routing layer. However, these dummy fills pose serious burden on the power/ground network when they are biased, while some of them may also be kept as floating. The floating dummies may also cause significant antenna effect that

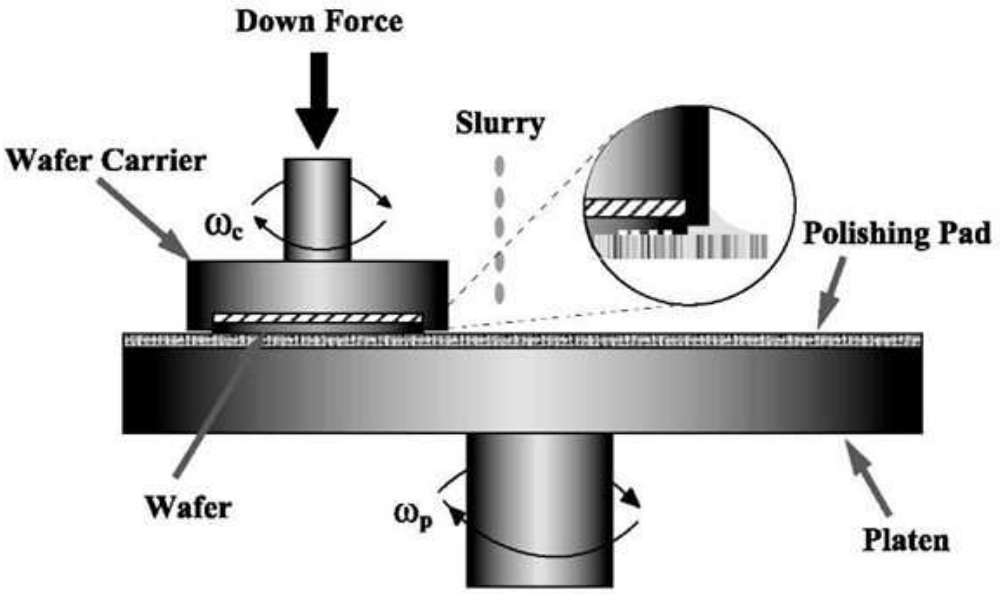


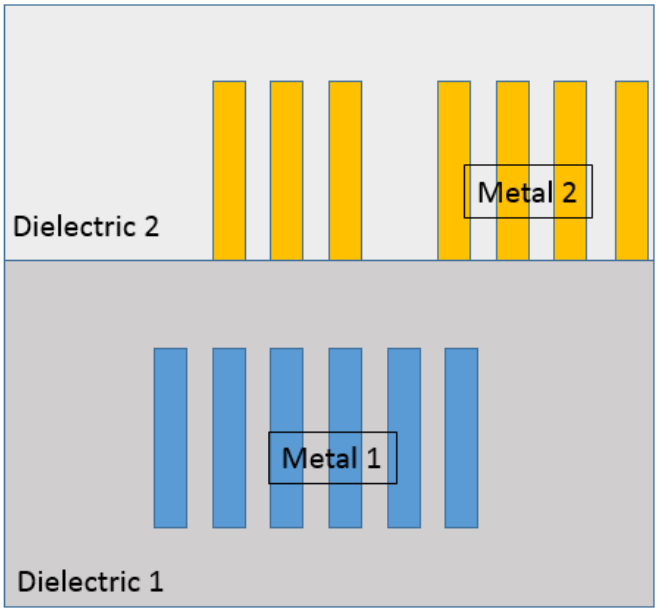
Figure 1.6: A schematic of Chemical Mechanical Polishing (CMP) Process [6]

may potentially damage the transistors due to large amount of static charge stored in them.

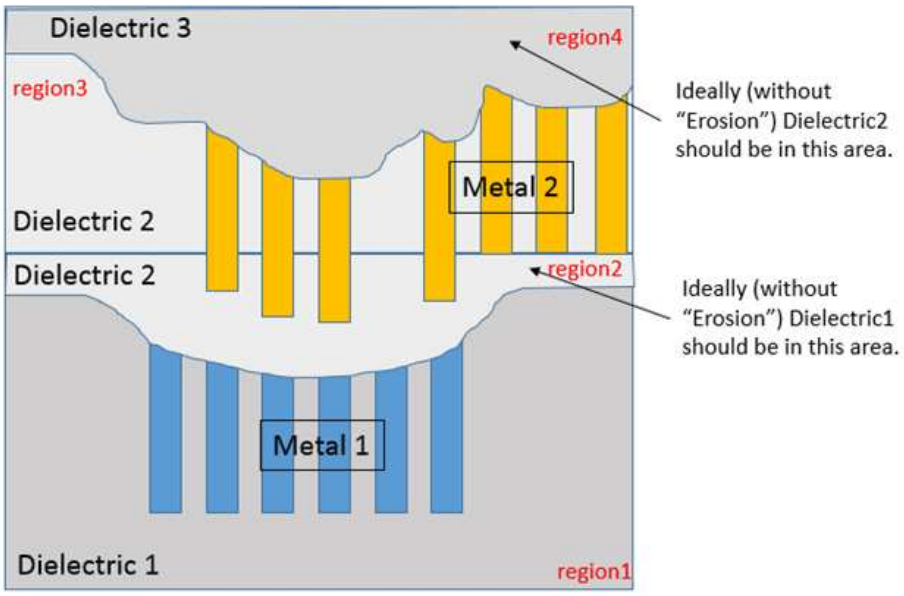
Beside the dummy fill approaches, recent academic research works have given more importance on uniform wire density distribution across the layout in a given metal layer, during global/detailed routing. In this approach, the idea is to minimize the number of metal fill by devising global/detailed routing methods while ensuring better uniformity in wire density distribution across the layout. If these issues are not given proper attention during the design process, an effective silicon implementation and the performance factors of a design under consideration may not be guaranteed. As a result, many of the fabricated devices may fail both functionally and structurally, giving rise to poor fabrication yield.

1.2 Reliability Issue due to Via Failure

In integrated circuit (IC) fabrication, the reliability of a design is also a major concern. Due to the growing integration demand in a single die, within a prescribed layout area, the complexity of a successful design closure increases aggressively. The fabrication processes allow multiple routing layers for 100% routing completion, incurring a very large number of interconnections between different segments of a net in different



(a)



(b)

Figure 1.7: Impacts of Chemical Mechanical Polishing (CMP) Process [7]: (a) ideal case, and (b) practical case

metal layers, known as vias. These vias can greatly impact the reliability of a design after fabrication, as a higher via count implies higher probability of via failure due to random variation induced by the fabrication process. A routing solution with

excessive via count not only leads to design for reliability issues due to random via failures, but also impacts the circuit performance due to increased resistance along the net having more vias.

Double via insertion during post-routing optimization or redundant-via aware routing optimization as depicted in Fig. 1.8 for increased reliability and manufacturing yield of the fabricated designs are some of the known approaches. Even redundant via aware engineering change order (ECO) routing during mask optimization are also being practiced toward the end of the design flow for further minimizing potential via failures.

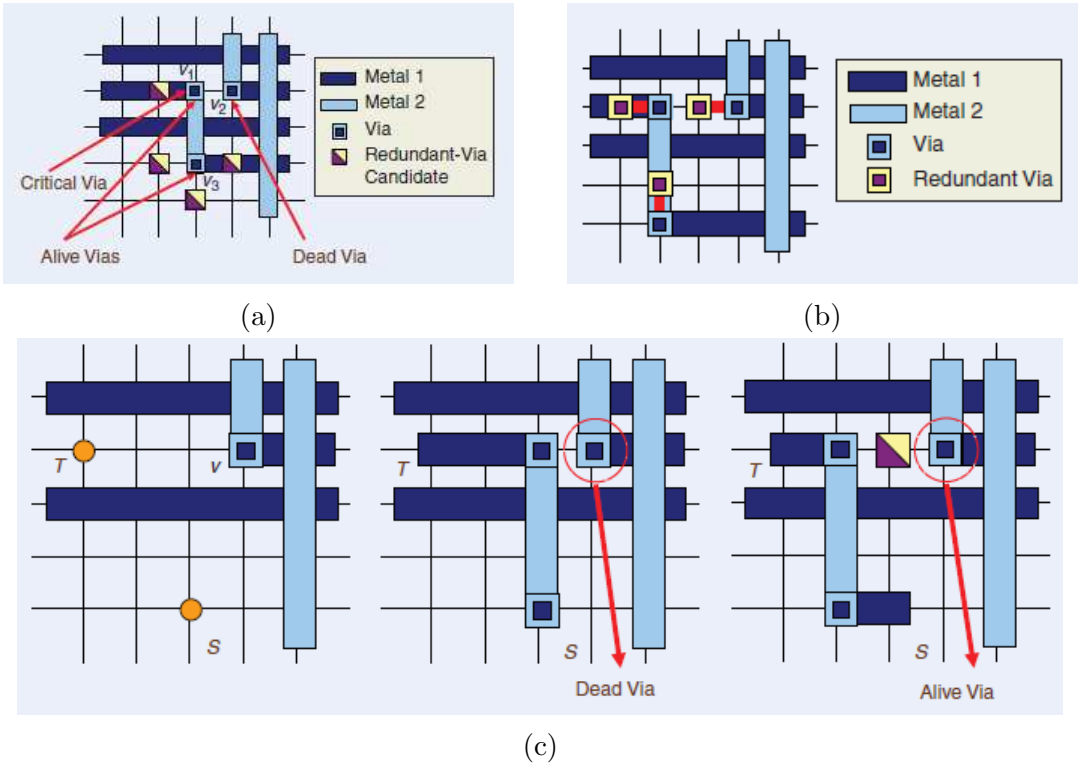


Figure 1.8: Design for reliability enhancement by redundant-via insertion and via-aware routing [2]

Since a via occupies a considerable amount of routing space, a design already having a large number of vias may not have the required space to accommodate an additional number of redundant vias for each of these vias in order to recover from the probabilistic failure of the original vias in the design. In many occasions, this mandates the entire design process to be redone or atleast conduct multiple iterations in global/detailed routing for an effective via minimization. Since global and

detailed routing are very hard problems to solve and mostly rely on constrained via minimization (CVM), a routing solution with minimal via count or even convergence of such a solution with 100% routing completion is hardly guaranteed.

1.3 Motivation for this work

In the existing PD flow as illustrated in Figure 1.1, global routing is an important stage after standard cell placement which is driven by congestion or timing or both. A global routing solution has significant impact on the quality and performance of a design, specifically ensuring a feasible detailed routing solution due to prevailing congestion scenario in the routing regions across multiple metal layers. Notably, the existing global routing framework is applicable only when the placement of the standard cells, the macros and the IO pads have been done with a specified design (placement) density. This will be an indicative of the adequate routing space for the given netlist, ensuring minimal/no design rule violations and fabrication issues as cited earlier. In summary, global routing acts as the pivot for the PD flow, first by evaluating the routability of a placement solution and secondly guiding the subsequent detailed routing of the nets for better manufacturability and reliability.

Based on the grid graph model, the existing global routing methods yield routing for a pair of terminals as a path between the centers of the corresponding routing bins containing those pins. The remaining routing from the bin centers to the actual pin locations, known as *pin-access problem*, is not done during global routing and traditionally being pushed to detailed routing. Moreover, this global routing framework also overlooks the nets that do not span between multiple routing bins due to the predefined bin size, by confining them to only one bin. These short nets are also not handled by the existing global routing and are handled during detailed routing. As a result, congestion estimation due to these local routing and the overall congestion estimation is not accurate. It is to be understood that if the overhead of the pin-access nets and the short nets within a routing bin is significant, the final routing solution can be prone to design rule violations for the very deep submicron fabrication process, both geometric and optical rules.

Another important aspect in the existing global routing paradigm is that the via count is minimized based on the initial planar ($2D$) routing solution. This $2D$ solution is then projected to multiple metal layers using some heuristic methods aimed at

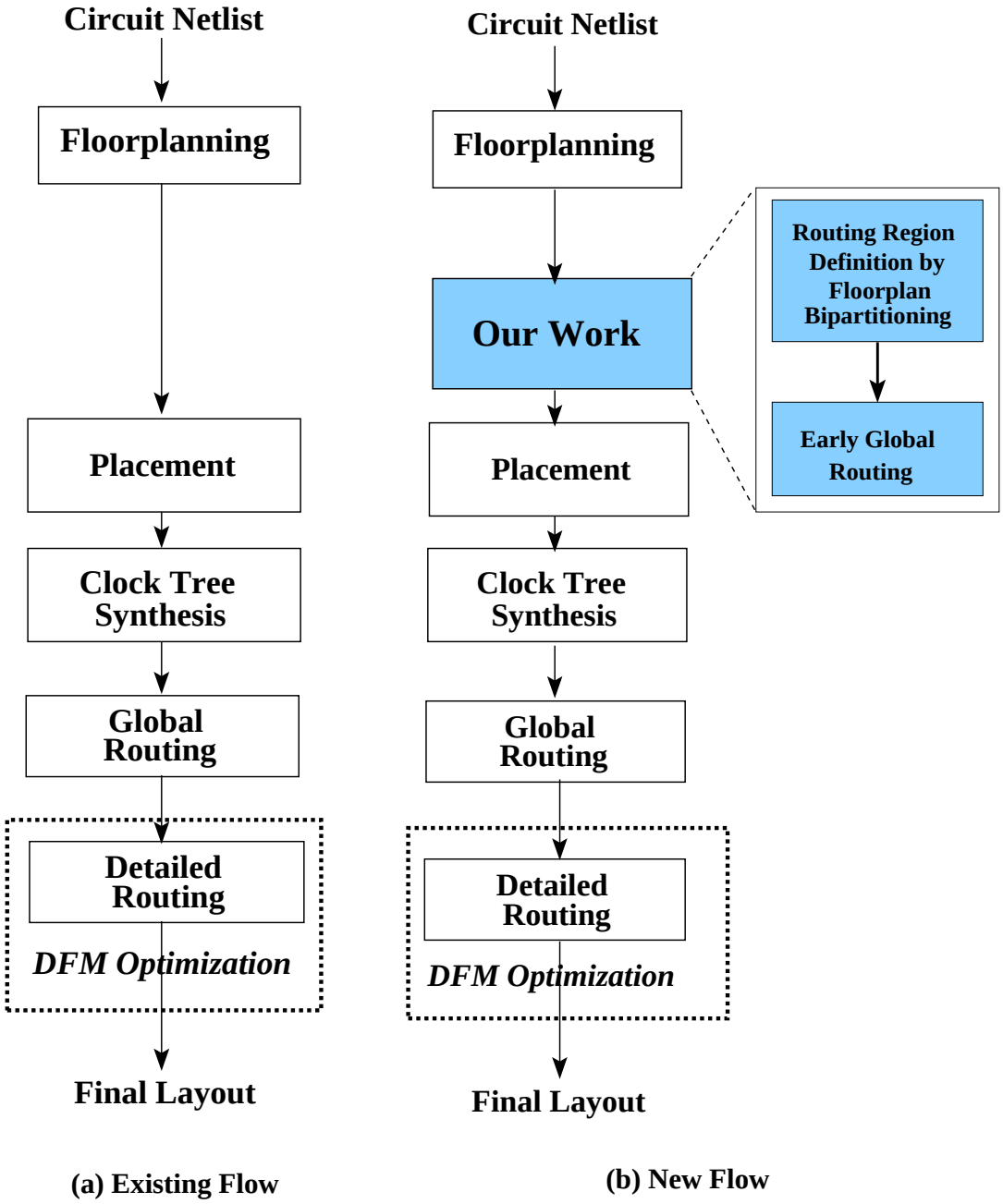


Figure 1.9: Overview of VLSI physical design (PD) flow: (a) the existing, and (b) the proposed aligned with this thesis

congestion mitigation, called constrained via minimization (CVM) approach without changing the planar topology of the net. This approach may have a huge impact on the reliability of the design due to large number of vias and their failure due to random process variation.

In Chapter 2, we conduct a comprehensive literature survey on the ongoing research issues, mainly focusing on the placement and routing optimization and DFM aware design approaches, in order to understand the current improvements in several physical design optimizating. This is very important for the design being implemented using VDSM nodes. Since the problems related to each of the physical design stages are very hard to solve, in addition to the design manufacturability and reliability issues due to VDSM technology based implementations, it requires multiple iterations at one or many stages of the PD flow. Sometimes, manual interventions are also needed when a very large number of iterations are not enough to bring a successful design closure with an acceptable design and performance criteria within a stipulated time frame. This study also highlights that these activities focused on earlier discovery of the problems found in later stages; leading to a successful routing solution with fewer DFM issues, yielding fewer lithography hotspots, pattern failures, more predictability in assessing the parasitic effects on signal integrity issues, and improved conformation for specified performance metrics of the design on silicon.

However, none of these optimization approaches did climb up the physical design flow (approaching from the fabrication end of the design flow towards system specification as per Figure 1.1) beyond the placement stage. Our study did not show any attempt was made at or immediately after floorplanning. This is due to the fact the floorplanning is done at a higher level of abstraction along the PD flow than the placement, therefore restricting the degree of information required by the existing global/detailed routing engines. Recently, a few floorplan bipartitioning methods pointed towards the possibility of performing early routability assessment on a given floorplan, without doing global/detailed placement of standard cells. They discussed about the potential of realizing an early global routing framework based on these bipartitioning results. Although there has not been any significant progress in that direction, by formulating a suitable routing model that can realize this prospect. A detailed summary of the literature survey is presented at the end of Chapter 2, highlighting the motivation for the objectives considered in this thesis.

1.4 Scope and Contributions of the Thesis

In this thesis, we explore a new outlook of the physical design flow intended to assess early routability of a floorplanned design. This is not feasible in the existing

global routing framework due to the limitations discussed earlier as well as in Chapter 2. Therefore, we propose an early global routing framework based on the floorplan bipartitioning information for defining the routing regions in the floorplan. This information is also used for computing the routing capacity for each of these regions, and identifying a well defined routing order of the nets. Finally, our aim is to adapt this early global routing framework with suitable abstraction of some of the design for manufacturability and reliability issues and assess the routability and other routing metrics at this early stage of the physical design flow. Our study on the existing physical design related academic or industrial research activities does not show the existence of any such early routing approaches on a given flooplan of a design.

1.4.1 Contributions

We summarize the following contributions made in this thesis:

Routing Region Definition by recursive floorplan bipartitioning in order to identify a set of monotone staircase routing regions in a floorplan. The nets abstracted at the floorplan level are to be routed through these regions in multiple metal layers. Unlike the existing maxflow based bipartitioning approaches, we proposed a faster floorplan bipartitioning framework, by simple graph search techniques on the floorplan topology graph.

The Early Global Routing Framework for routing of a set of nets in a floorplanned layout. Unlike the existing grid graph model, we propose the proposed early global routing framework that restricts the routing congestion in these routing regions to 100% and gives an overview of the global congestion scenario in the floorplanned layout. In this routing model, the scope of addressing the pin access problem at the floorplan level was also discussed.

Early Via Minimization by identifying a set of monotone staircase routing regions with minimal number of bends by an improved framework for recursive floorplan bipartitioning. This is an earlier version of the unconstrained via minimization (UVM) problem.

Exploring the scope of DFM Awareness and enhancing the proposed early global routing method by:

Uniform Wire distribution in the floorplanned layout in and across multiple metal layers for reducing surface irregularities due to CMP process in the final layout,

Minimizing Lithography Hotspots due to EPE by suitably modeling EPE effect into the routing cost of the proposed early global routing framework, using the simulation results obtained during the post-layout optimization, and

Over-the-Block Early Global Routing Model in order to facilitate routing of the nets over the blocks in higher routing layer, extending the proposed monotone staircase based routing framework by incorporating the existing grid graph model with suitable adaptation.

Integration and Validation with Industrial Design Flow for a comparative study between the traditional and the proposed physical design flow (see Figure 1.9) is conducted with the help of an industrial physical design tool on a few floorplan instances of a given industrial design. In this exercise, the proposed early global routing framework is integrated with the industrial physical design tool for exchanging relevant information between the proposed method and the industrial PD tool. The results obtained show reduced worst case congestion, similar average congestion and smaller runtime values, while marginal increase in wirelength and via count were reported.

1.5 Organization of the Thesis

The organization of the rest of the thesis is as follows:

Chapter 2 gives a detailed literature survey on the relevant academic research works pertaining to the existing physical design flow focusing on global (and detailed) routing and its dependency on the earlier stages such as placement and floorplanning; this chapter also discusses the scope of addressing design for manufacturability (DFM) issues such as optical proximity errors (OPE), edge placement errors (EPE) and the problems related to chemical mechanical polishing causing surface irregularities.

Chapter 3 presents a faster recursive flooplan bipartitioning method for identifying a set of monotone staircase routing regions in a floorplan of a design for the

proposed early global routing framework. In this chapter, a heuristic method employing BFS traversal on the flooplan topology graph is presented followed by a variant that uses DFS based graph traversal on the same floorplan graph.

Chapter 4 proposes an early global routing framework using the monotone staircases as the routing regions in order to estimate the routability of the given floorplan of a design. A set of nets abstracted at the floorplan levels are routed through these routing regions using a number of metal layers, by employing both unreserved and reserved layer model for routing. This chapter also presents a few extensions of this early global method, suitable for newer fabrication processes and different approaches for obtaining a routing path between a pair of net pins.

Chapter 5 discusses how unconstrained via minimization (UVM) problem can be addressed during the proposed early global routing by recursively identifying a set of monotone staircases with minimal number of bends, one by greedy approach and another by a randomized approach.

Chapter 6 discusses how we address the objective of uniform wire distribution in the proposed early global routing framework. This chapter also presents a new hybrid global routing framework using the framework proposed in Chapter 4 and an early adoption of the existing grid graph model used in the existing post-placement global routers. In this hybrid framework, we incorporate the routing penalty due to an early abstraction model of edge placement errors (EPE) due to the limitations of the existing lithography system.

Chapter 7 presents a case study of an industrial design in lieu of the existing physical design flow and the proposed physical design flow illustrated in Chapter 1.

Chapter 8 summarizes the works presented in the thesis and indicates some of the future research directions based on this work.

Chapter 2

Literature Survey

In this chapter, we present the survey on the existing literatures pertaining to the physical design (PD) flow, focusing on the placement and routing optimization, approaches to address design for manufacturability (DFM) issues [18] and chemical mechanical polishing process [6]. We also extend this study for via minimization approaches during the post-placement routing phase, since an excessive via count can risk the reliability of the design being fabricated into the silicon in smaller technology nodes. We also pay attention to the design styles in the context of large ASIC/SoC designs where large number of pre-designed hard modules (macro blocks) are imported in the final product along with a set of pre-designed soft blocks containing a number of macros and standard cells. Our study also explores the existing literatures related to the feasibility of early routability and congestion assessment on a given floorplan, earlier than the existing practices as per the current PD flow.

2.1 The Place and Route Framework

In the physical design (PD) flow depicted in Figure 1.9 (a), global routing is an important step after placement and guides the subsequent detailed routing addressing the manufacturability and reliability issues prevalent in VDSM process nodes such as $45nm$ and below. Industrial such as Olympus-SoC [13] as well as academic global routing methods such as [19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32] strictly follow the existing PD flow. In this global routing framework, grid graph model is used for obtaining the routing path of a net over a set routing (metal) layers. This graph is constructed by dividing the layout area in equal sized bins, called *global*

routing bins or GCells. Initially, a planar ($2D$) routing solution is obtained for the bottom most layer pair, i.e., (M_1, M_2). Subsequently, layer assignment of the nets in different routing (metal) layers beyond M_2 is done while minimizing the number of vias and the congestion [22, 23, 24, 33, 34] in the planar routing solution. If this solution is not able to attain 100% routability or the congestion in the routing regions exceeds 100%, ripup and reroute (RR) method is applied to those failed nets. In this regard, the routing order of the nets plays a very important role for identifying a feasible routing solution with 100% routability, fewer vias and congestion hotspots. This solution enables the subsequent detailed routing for obtaining an acceptable final routing solution that is free from design rule check (DRC) violations and DFM issues due to the limitations of the lithography system for VDSM technologies discussed earlier. Otherwise, several iterations of post-layout optimization are to be carried out. In many cases, this approach does not prove to be effective for attaining a routing solution free from the said violations while obeying the performance specifications. Several iterative approaches for DFM optimization are being practiced in earlier stages such as global routing, placement etc. in order to refine the respective solutions for a violation free acceptable design with desired performance metrics.

2.1.1 Design Styles for Physical Implementation

Traditionally, there are two different design philosophies known to the designers. Depending on the size of the design, either a hierarchical or a flat design methodology is adopted [13, 17]. In the flat design framework, each constituent functional blocks are flattened at the top level in order to obtain a flat netlist. This flat netlist is then used as an input to the subsequent design stages, such as physical design steps. On the other hand, with increasing design functionality in today's semiconductor industry, the component functional blocks are themselves quite large in terms of the number of components that are estimated to contain. As a results, these component functional blocks are designed separately, both as soft blocks and macros, so that they can be used in one or more parent designs with fewer difficulties. This is hierarchical design approach.

In hierarchical design style, first all the physical design optimization are done at the component level in order to generate the corresponding macro blocks. These blocks are made to comply with all the design sign-off rules such as fewer/no design rule violation, no scope lithographic hotspots, etc. On the contrary, soft blocks, each

being viewed as a cluster of standard cells, comes as functionally verified components. Their physical design implementation need to be handled by the top level optimization tools along with available macros and pads. In the existing PD flow, the floorplan optimization is done based on the macros, IO pads, and the estimated are of the soft blocks. Subsequently, a detailed placement solution is obtained based on this floorplan topology and the other constraints like timing, routability, initial congestion. A complete placement solution with legally placed macros/pads and standard cells is then considered for formulating the existing global routing framework.

2.1.2 Interleaving Placement and Routing Methods

During placement stage, predicting a feasible placement solution for better routability [35] and congestion scenario [36, 37] is a very hard problem. Notably, congestion estimation for a given placement solution is not reasonably accurate if the routing of the nets is not done. Integrated placement [37, 38, 39] framework employing a faster yet effective global routing tool suite FastRoute [21, 27, 40, 41] is one such effort toward a faster routability assessment on a placement solution. Using this integrated framework, the quality of a placement solution can be assessed very quickly in terms of the critical routing parameters such as routability and congestion. Few other routability driven placement methods like IPR [42], POLAR 2.0 [43], Ripple 2.0 [44] were also proposed for an acceptable placement solution with a good global routing solution. Moreover, integrated global and detailed routing solution such as GDRouter [45] and full-chip multi-level routing frameworks employing global and detailed routing together such as MARS, MR, MGR [46, 47, 48, 49], DUNE [50] etc. were also proposed for better routing completion for a given placement solution.

In the multi-level routing frameworks, both global and detailed routing of the nets are done at different levels, during coarsening and uncoarsening of the grid graph. The coarsening procedure continues until a threshold value for the size of GCells is reached, in order to route the nets of smaller bounding box to the larger ones in the corresponding levels. Once the coarsening process stops at some point, uncoarsening process starts to refine the routing results till the level 0 is reached, by suitably ordering the nets and employing rip-up and reroute repetitively. Multilevel routing frameworks were also explored in optimization of OPC [51] as well as for CMP variation control [52] for better handling of the corresponding DFM issues in post-layout optimization as well as in testability and yield enhancement [53]. Unlike others, [52]

used the uncoarsening stages first followed by coarsening stage for better distribution of metal density across the layout. However, they are very time consuming and efforts have been made to reduce the time complexity [54].

2.1.3 Pattern Routing Frameworks

For faster global routing methods with predictable solutions, the idea of pattern based routing [19, 20] came in to existence. Until then, Maze routing [55] had been an effective method for older global routers before. In general, pattern routing [20] using L/Z (one/two bend) or even monotone staircase (multi-bend) patterns [19, 27] have become more popular than the variants of maze routing methods [55]. In pattern routing, multi-bend monotonic pattern routing is the most flexible variant among all the routing patterns in identifying a routing path, despite numerous routing blockages within the net bounding box. However, there are cases when the routing of the nets can not be confined within the respective bounding boxes due to excessive routing blockages and the overall congestion scenario due to already routed nets. In FastRoute-4.0 [27], in addition to the above mentioned patterns, a predictable three-bend (U) routing pattern was proposed that is slated to incur fewer number of vias than traditional maze routing based detours. These routes are shown to be beneficial in case of detoured routes with fixed number of vias with a similar cost of increased wirelength like detoured maze routing. In many cases, the detours using U patterns may lead to inferior routing solutions with unacceptable increase in wirelength and thus leads to its reduced usage. Moreover, the authors in [20] have also shown that pattern routing is potentially favorable in case of signal integrity issue by reducing excessive parasitic effects on the signal nets due to nearby clock nets. On the other hand, maze routing and its variants such as $3D$ maze routing [22], $A*$-search based algorithms [17, 56] are the most suitable method even today when the pattern routing with one, two or more bends fail due to heavy routing blockages within the net bounding box in the lower routing layers. Notably, this comes at the cost of higher runtime [19] than the pattern routing frameworks.

Although multi-bend monotone staircase patterns are the most flexible ones, they incur more number of vias along the routing paths as compared to L/Z patterns with fixed one or two via overhead. In the existing global routing framework, after the planar solution is obtained, the net segments are moved to the higher metal layers for congestion mitigation. This is called layer assignment problem [17, 56]. Several

methods have been proposed in order to minimize congestion [23, 33, 57] during the global routing mostly by greedy heuristic, while probabilistic approaches [58, 59, 60] were also considered for pre-routing congestion estimation. During layer assignment step, a set of vias are used for the interconnection of the net segments belonging to the same net in different metal layers, without changing the routing topology obtaining in planar routing solutions. This is called constrained via minimization (CVM) problem and mostly prevalent in the existing global routing methods. One such example is FastRoute-4.0 [27] that relied on via aware Steiner tree topology generation for initial planar routing solution. While BFGR [24] used via minimization by some empirical via pricing, NTUgr [31] highlighted dynamic programming based $2D$-to-$3D$ layer assignment of the nets and via sharing among the common net segments. On the contrary, very few efforts have been made toward a framework that facilitates simultaneous via minimization while finding a routing path between a pair of terminals across multiple metal layers. This is commonly known as unconstrained via minimization (UVM) problem [17, 56, 61]. To some extent, the authors in FGR [22], MR [48] talked about $3D$ maze routing for better routing results with fewer via count while LMGr [30] attempted to address via minimization by estimating the bends along a routing path.

2.1.4 Multi-Terminal Net Decomposition

So far, we have studied the global routing paradigm generalized for a set of nets without any specific number of terminals (pins). While routing a two terminal net is just to obtain the routing path among these two pins only, specifically center-to-center routing between the GCells, routing of multi-terminal nets requires additional treatment before the routing of these nets is obtained. The optimal routing of a multi-terminal net is a NP-hard problem, so typically it is decomposed into a set of valid two-terminal net segments. The routes of these segments are identified subsequently. There exists two different frameworks for multi-terminal net decomposition based on: (i) Steiner tree, and (ii) Minimum Spanning Tree (MST). Since the routing is allowed in the preferred routing directions only, either horizontal or vertical, rectilinear Steiner/Spanning tree based decomposition is adopted. A rectilinear Steiner tree based net topology with minimal wirelength, abbreviated as RSMT (rectilinear steiner minimal tree), is proven to yield better routing solutions than the rectilinear minimum spanning tree (RMST).

For the RSMT framework, first a set of vertical and horizontal lines are projected from each pin of a net. These lines form a grid called *hanan grid* [9]. The intersection points of this grid denote the locations of the pins as well as possible locations of a special set of points called *Steiner points*. Since the pins are not equally spaced, the width of the adjacent grids are not the same and hence the overall grid structure is not uniform. For ease of understanding and computation, the spacing between each grid line is assumed to be uniform and is the basis of the recent methods for Steiner tree topology generation methods such as FLUTE [10]. Most of the recent global routing methods use RSMT based multi-terminal net decomposition, while a few academic routers like FGR [22], NTUgr [31], BFGR [24] used RMST based decomposition for multi-terminal net routing. It is known that the Steiner tree problem [62] is very hard and several heuristic and approximation methods have been proposed [17, 56]. Since the routing of the nets are done along horizontal or vertical tracks, Rectilinear not Euclidean Steiner Tree is applicable. In rectilinear domain, *Steiner ratio*, defined as the ratio of the total length of the minimum spanning tree to the minimum Steiner tree, has a maximum value of 1.5 [63].

2.1.5 Limitations of Grid Graph Model

The fundamental limitation of the grid graph routing model is to decide the size of the global routing bins (GCells). In the existing global routers, the size of these bins are predefined. While a larger bin size facilitates faster routing time, it gives poor routing results and skips many of the nets falling entirely within a bin. On the other hand, a smaller bin size routes more nets and thus needs more routing time. Despite that, there can be many nets that entirely falls within these smaller sized bins and remained unrouted. In fact, any net that falls entirely within a GCell is not considered to be routed by the global routing engines. Instead, these are pushed to the detailed routing stage. Theoretically, it requires the bin size to be zero in order to route all the nets by global routing methods. The routing path of a net connecting two pins located in two different GCells is identified as the path that connects those GCells, terminating at their respective centers. The path from the center of a GCell to the actual position of the pin located within this GCell is not obtained at this stage of routing. Additionally, the nets that sit entirely within a GCell are overlooked and pushed to the detailed routing stage. As a result, inability to address the routing of these nets during global routing leads to a critical problem

known as *pin access problem* [64, 65]. The pin access problem is a huge bottleneck for achieving routability and fewer congestion. The local net segments used for pin access and short nets significantly consume available routing tracks that can otherwise be used for long (global) nets running across the GCells. A score based approach for ensuring routing the pin access nets and the short nets within a GCell was presented in [66] for maximizing routing track utilization for both intra-bin local net routing and inter-bin long net routing. Handling the pin access problem during detailed routing along with an improved global routing solution is the main reason behind the recent approaches of integrated global and detailed routing such as GDRouter [45] and multi-level full chip routing frameworks such as [46, 48, 49].

Since these local (intra bin) nets are not routed during global routing, the local congestion within a bin due to these nets are very hard to estimate. The local congestion within a bin in many cases can cause serious problems during detailed routing, creating large number of congestion and lithography hotspots. Design rule check (DRC) may fail when the net estimate during global routing do not show any congestion issues due to these local routing segments used for pin accessibility. Although a few works [15, 64] attempted to address this issue by a heuristic technique called resource reservation for local routing and congestion, they did not provide a reasonable solution due to the inherent limitation as cited earlier. In another instance, a global routing solution and hence a placement solution fail to cater a feasible routing solution when more number of nets terminate within a GCell than the number of routing tracks available on its boundary edges, causing routability driven placement and physical synthesis to fail mitigate the failure due to do intra-bin routing and in accurate local congestion estimation [65]. A pin density based method GLARE [15] tried address this local routability issue. In case of failure, a completely new or iterative placement strategies are to be adopted [21, 38] such as global swap or movement along the placement row followed by subsequent global/detained routing [13]. Since there is no guarantee on the optimality of the recurring placement solution, a large number of iterations may be required before successful design closure. This is a huge bottleneck for highly competitive design time-to-market for the present semiconductor industry.

2.2 DFM and Reliability Issues

In modern IC fabrication process nodes such as those below $65nm$, severe yield loss has been observed that are primarily caused by design for manufacturability (DFM) issues [67]. For integrated circuit (IC) design using very deep submicron (VDSM) technology, such as $65nm$ and below, the design rule check (DRC) process is very crucial. Using standard layout design rules meant for higher technology nodes such as $90nm$ is not sufficient due to the increasing number of complex design rules arising out of design for manufacturability (DFM) issues prevalent in lower technology nodes. Due to the limitation of the existing optical illumination system using $193nm$ wavelength in IC fabrication process, it has become too troublesome to handle sub-wavelength feature sizes ($\ll 193nm$). This makes several design for manufacturability (DFM) issues such as optical proximity errors (OPE), edge placement errors (EPE) critical, causing structural as well as electrical faults [2, 68, 69] in the design.

Even in analog layout design with smaller technology nodes, with fewer design components than their digital counterparts, OPC and EPE effects can heavily degrade the performance factors and require special care to alleviate OPE/EPE issues to large extent in reasonable time [70]. Since the number of patterns (rules) needed to alleviate OPEs in design are increasing exponentially with the newer nodes, simulation based paradigms such as quasi-inverse lithography method [4] and model based initial biasing method [71] are proved to be accurate and conducive for faster convergence during routing/post-layout OPC optimization. Of course, there lies a very large overhead of simulation time, being mostly one time cost.

2.2.1 Resolution Enhancement Techniques

Several resolution enhancement techniques (RETs) like rule/simulation based optical proximity correction (OPC) techniques [2, 18, 67], phase shift masks (PSM) [18, 72, 73, 74], off-axis illumination (OAI) [18, 75], immersion lithography [18, 76, 77, 78], Extreme Ultraviolet (EUV) [18], Electron Beam lithography (EBL) [18, 79], multiple patterning such as double/triple patterning lithography (DPL/TPL) [80, 81, 82] etc. came into existence to alleviate the limitation of the existing illumination system [2]. These methods are usually applied after the traditional physical design flow is completed and also needs to be verified by the standard physical verification tools, e.g., Calibre $nmDRC$ tool suite [83, 84] that performs both traditional design as well

as optical rule check (ORC). These issues are mostly considered towards the end of the physical design flow such as during post-layout optimization or in detailed routing, in addition to a few rule based OPC aware maze routing [51, 85, 86], simulation based OPC aware routing [4] and post-routing optimization [87], EPE aware detailed routing framework [12, 88, 89].

2.2.2 EPE aware Optimization

Notably, few OPC or EPE aware global routing frameworks exist in order to address them as early as possible to avoid too many design iterations. Simulation based OPC [4] framework employed quasi-inverse lithography based predictive formula for OPC in order to identify the deficiency in optical intensity within the expected metal contour and guided the subsequent detailed routing for minimizing such deficiencies causing OPEs. On the other hand, EPE is a result of the failure to get the metal contour printed properly, as a result of intensity spreading beyond the line contours due to diffraction effect at the edges of the mask opening. This spreading can go beyond the contour line where the intensity drops to 30% of the maximum intensity at the middle of the mask opening.

A fast simulation based EPE-aware detailed routing method was proposed in RADAR [12]. In this work, both wire spreading and ripup and reroute based techniques were used for minimizing EPE hotspots in the layout. In ELIAD [88], an EPE aware detailed routing method was presented based on statistical characterization of known litho-prone shapes such as jog-corners, vias and line-ends. However, the results on a couple of industrial designs presented in this work considered only two metal layers and identified EPE hotspots based on some predefined threshold values. Another EPE aware detailed routing method AENEID [89] used kernel based data learning for EPE hotspot detection and routing path prediction. These kernels were trained and validated beforehand on the existing simulation results on a large number of litho-prone patterns from different designs. However, the results were shown for two metal layers only, similar to [88], and both are post-OPC methods. Due to the lack of suitable model for OPE, EPE as well as other RET issues available at the early design stages, it is hard to predict these failures and also incorporate in early routability assessment. In this thesis, we intend to focus on realizing the models for EPE and CMP effects in a design during early global routing after floorplanning.

2.2.3 CMP aware Optimization

Another DFM issue that greatly impacts the design quality is chemical mechanical polishing (CMP) which induces surface irregularities during the copper metalization process due to different hardness factors of copper and dielectric materials and also the variation in wire (metal) density across the layout. As a result, over-polishing and under-polishing of these materials occur across the metal layers [90] causing: (a) structural faults like open and shorts due to lack of sharpness around depth of focus (DOF) of the lithographic illumination system, and (b) the unpredictable electrical characteristics of the metal wires such as resistance and capacitance that impact both power and timing. So far, insertion of dummy metal fills [91, 92] has been a popular method to alleviate these issues as these tend to (i) reduce the unpredictability in the electrical properties of the metal interconnects, and (ii) improve uniformity in metal density across the layers for less CMP variations.

But, these dummy fills pose serious burden on the power/ground network due to induced cross-talk from high coupling capacitance and also due to IR drops [93]. Therefore, a uniform feature density distribution including an effective placement of cells and pins [94, 95], and suitable global [96, 97] or detailed routing [52, 91, 92] solution that yields uniform distribution of the metal interconnects across layers are more practical solutions than mere dummy metal fills [91, 92, 98].

2.2.4 Via aware Reliability Optimization

Another critical aspect of VDSM fabrication technology is the reliability of the fabricated designs due to large via count [2]; via fabrication may yield poor results due to statistical process variations so that vias may fail to connect electrically or may impact electrical performance due to higher resistance/capacitance. Therefore, it demands significant number of redundant vias to be placed around a target via and a failure to do so may entitle to failure of the entire chip to function or give poor performance. Even, no industrial tools exist to address this issue earlier in the PD flow for a faster and successful design closure. Double via insertion during post-routing optimization [99, 100] or identifying a via-failure aware routing or even redundant via aware ECO routing during mask optimization [101] for increased reliability and yield of the fabricated designs are some of the known approaches. In summary, in order to get better manufacturing yield, the design flow needs to account for design for reliability issues due to via failures and to put effort in minimizing the number

of vias as early as possible, during global or detailed routing instead of post-routing redundant via insertion or ECO based approaches.

2.3 Early Routability Assessment on Flooplans

In order to predict the feasibility of a routing solution due to an inferior placement solution, an *early global routing estimation* on a given floorplan of a design for the targeted placement solution may potentially help, by defining routing regions in a floorplan with the help of recursive bipartitioning methods [8, 17, 102, 103]. This can be realized without doing a full fledged global routing on a given placement solution. This is due to the fact that a feasible floorplan solution can be endorsed by a good early global routing solution, thus reducing the number of iterations for the subsequent global/detailed placement of standard cells only. This early global routing solution will guide global routing of the remaining nets connecting the standard cells. Notably, the nets at the floorplan level defines the connectivity among the macros, IO pads, and soft blocks (a cluster of standard cells). The main difference is that these abstracted nets terminate on the boundary pins of the macros, pads and the soft blocks. Although, the soft blocks do not have any physical pins at their boundary, the concept of virtual pin can be used [13]. The virtual pins at the boundary of the soft blocks will guide the subsequent global routing of the remaining segment (not routed by this early global routing framework) of the particular net entering the soft blocks and terminates on a standard cell within it. In other cases, some of the nets may fully be contained within a soft block and have not been routed by the early global routing framework can be handled entirely by the existing global routing methods. But, no such early routability assessment methods are known to exist till now and therefore is the main objective of this thesis.

2.3.1 Advantages of Monotone Staircase Cuts over Slicing Method

In a floorplan, a set of routing regions can be defined by recursively bipartitioning the floorplan either by isothetic cut lines for sliceable floorplans [17] or by monotone staircase cuts [8, 102] for any floorplan irrespective of its sliceability. While slicing method fails to obtain a bipartition on a non-sliceable floorplan, monotone staircase cuts can obtain a number of possible bipartitions of any floorplan irrespective of its

sliceability. Notably, the monotone staircase cuts based bipartitioning framework is a generalization of the slicing tree method. The corresponding cuts yield the required routing regions for identifying an early global routing solution of a set of floorplan level nets, a subset of the placement level standard cell netlist. The net cut information obtained at each level of this recursive procedure yields the routing capacity of the respective regions. It is also known that a good (acyclic) routing order is very important for 100% routing completion [17, 104, 105], without too many ripup and reroute effort during post-placement routing optimization.

In [17, 105], it was shown that non-sliceable floorplans do not cater acyclic routing order, such that the nets can be routed one-by-one in a definite order. On the other hand, monotone staircase based recursive floorplan bipartitions provide a *tree like* (acyclic) routing order irrespective of the sliceability of the given floorplan [104, 105]. Furthermore, the monotone staircases provide a certain degree of flexibility in resizing the regions bounded by these staircases for efficient resource utilization, by incremental placement of macros/cells. This would create the required routing space in order to mitigate excessive routing congestion in those regions. A judiciously chosen monotone staircase routing region with minimal number of bends and minimal net cut can potentially help in finding a routing path between a pair of terminals across the metal layers, with less congestion and fewer via count along the routing path.

2.3.2 Existing Bipartitioning Methods

Until now, there exist different variants of minimum net cut based floorplan bipartitioning methods focusing on *area balance* or *number balance* between the partitions [8, 106]. The area balanced bipartitioning problem is known to be NP-hard [8] while the number balanced version is a special case of it. Previous works on area balanced bipartitioning [8, 102, 106] employed iterative maxflow-based heuristic methods and handled two ($t = 2$) and multi-terminal nets ($t > 2$) differently with different heuristic approaches. In [107, 108], it is initially shown that a circuit graph can be bipartitioned by an iterative maxflow based node-merging technique. This algorithm used an $O(n^3)$ push-relabel algorithm [109] for computing the maxflow in each iteration leading to $O(n^4)$ time for a maximum of $O(n)$ iterations. Further improvement was also suggested in [107] as iterative flow augmenting procedure similar to Ford-Fulkerson algorithm [109] with $O(nm)$ time complexity, where n and m respectively denote the number of vertices and edges in the corresponding floorplan graph. In summary, the

existing maxflow based area/number balanced bipartitioning methods take huge time.

A linear time algorithm for *number balanced* floorplan bipartitioning was proposed in [103] by employing depth-first traversal on the given floorplan channel graph. This method, namely *Block_Labeling*, identifies monotone staircases by labeling the blocks one by one until $\lfloor n/2 \rfloor$ blocks are labeled, constituting the left partition. However, this method did not consider net cut as an objective, nor is adaptable for area balanced bipartitioning as a general case. In summary, it requires a faster hierarchical floorplan bipartitioning method that gives monotone staircases with minimal net cut as well as unique method for handling nets with any number of pins. As discussed earlier, it is also required to obtain a set of monotone staircase routing regions that will facilitate minimal bend pattern routing through them for addressing UVM problem at the floorplan abstraction level.

2.4 Chapter Summary

From this study, we notice that routability and congestion analysis is done during the traditional post-placement global and detailed routing stages, for the respective global and local congestion. In many cases, the criteria for meeting the critical performance metrics may lead to multiple congestion hot-spots, leading to proportional increase in lithography hot-spots. The number of vias due to multi-layer interconnection among the wire segments belonging to a given net can only be obtained only after routing is completed. These are very crucial parameters that demand minimization during the design implementation phase, specifically the placement and routing. Little attention was paid in order to assess the impact of these issues at an earlier stage such as during or after floorplanning. As a results, multiple iterations are needed at different stages, such as during routing and placement, making it very hard to meet the stipulated design time with desired design objectives. Despite the interleaved placement/routing and multi-level routing frameworks as well as advancement in faster Steiner topology generation, and Steiner topology, pattern routing and congestion aware CVM approaches were proposed; design closure os not necessarily attained with desired success with minimal fabrication related issues. While manual intervention with extensive design exposure has been a naive approach, starting over with a completely new solution from an earlier stage such as floorplanning and placement stage is also being attempted. Given the scale of design complexity and large number of functional

blocks in modern ICs, this is a huge bottleneck to stick to a competitive design time to market.

From this discussion, we summarize that an early global routing framework based on floorplan information is essential for studying the routability of a design much earlier in the PD flow, i.e., immediately after floorplanning than the usual practice of post-placement routability assessment. In addition to that, an early estimation of via count and congestion within the routing regions defined for a given floorplan topology can be a decent indicative of the quality of the design at the floorplan abstraction level, with potentially fewer iteration, than during post-placement routing or DFM aware placement/routing or even during post-layout DFM optimization.

Therefore, in this thesis, we intend to address the following by proposing a new framework for early global routing immediately after floorplanning:

- obtain an estimate of routability, wirelength, via count and congestion at this early abstraction design level, since no such framework exist till date;
- present a method for addressing an earlier version of unconstrained via minimization (UVM) problem for simultaneous multi-layer routing topology generation and via minimization, which is hardly being used in the existing post-placement global routing approaches;
- explore the scope for alleviating pin access problem at floorplan level by suitably defining the pin-region edges in the proposed routing model, a critical problem still prevalent in global routing due to the inherent limitation of the grid graph model;
- perform routability assessment of the designs considering suitable abstraction of DFM issues at such an early stage of PD flow, alike a few DFM aware post-placement routing methods; and
- cater an early global routing solution as an initial solution to the subsequent placement and routing optimization, specially for large mixed mode design environment and hierarchical design practices, for reducing the number of iterations at the subsequent stages of PD flow.

Chapter 3

Fast Recursive Bipartitioning for Early Global Routing

3.1 Introduction

In the existing floorplan optimization tools, an optimal floorplan is identified as the one with minimal cost in terms of half perimeter wirelength (HPWL) of the nets. Obviously, this floorplan instance does not necessarily guarantee a good placement solution with 100% routability, reasonably small wirelength and fewer congestion hotspots. Hence, the impact of the corresponding placement solution on the routing results is discovered quite late as per the existing PD flow, while conducting design rule check (DRC) and discovering the issues related to design for manufacturability and reliability. In order to alleviate these fabrication issues, several iterative efforts are put in detailed/global routing, global/detailed placement of the standard cells and even in floorplan optimization. So far, there is no direct way to measure whether a (primitive) floorplan instance is capable enough to route many of the nets abstracted at the floorplan level or to infer that the corresponding placement is able to yield an acceptable routing solution with minimal or no violations. Since, the placement details of the individual standard cells are not available at this stage, the existing global routing methods can not be used so early, i.e., after floorplanning in order to assess the routability and other routing metrics in lieu of the corresponding floorplan instance. Additionally, the netlist abstracted during floorplan optimization also differs grossly from the netlist comprising of standard cells and macros used during placement optimization stage, as the connectivity of the standard cells are masked.

In this thesis, we propose a framework for early estimation of the routability and other routing metrics such as wirelength, via count and congestion, much earlier than the existing PD flow as depicted in Figure 1.9 (b). In order to achieve that, our proposed early global routing framework comprises of two sub-stages, namely: (i) identify the routing regions in a given floorplan using recursive floorplan bipartitioning methods, and (ii) formulate a new global routing model using these routing regions and identify the routing topology of the nets abstracted at the floorplan level. The bipartitioning results also help this early global router identify an well defined routing order of the nets and also computes the routing capacity for each of these routing regions.

3.2 Floorplan Bipartitioning

The most commonly known floorplan bipartitioning method uses slicing tree method in order to identify the routing regions adjoining the block boundaries, using only vertical or horizontal cut lines. But this method is not applicable to non-sliceable floorplans (see Figure 3.1 (a) and (b)). Recently, a new paradigm for floorplan bipartitioning was proposed in [8, 102, 103]. These methods identify the routing regions in any floorplan irrespective of its sliceability, by using monotone staircase cuts, in a recursive manner. A hierarchy of bipartitions for a given floorplan, similar to slicing tree method for a sliceable floorplan, is obtained by these recursive methods. A set of objectives were defined for obtaining an optimal monotone staircase cut at each level of the bipartition hierarchy. The corresponding routing regions take the shape of a monotone staircase spanning from one end of the floorplan (or the subfloorplan at a given bipartition hierarchy) to the other end. The illustration in Figure 3.2 shows the enumeration of a monotone staircase as a routing region in a floorplanned layout [8].

In the next section, we discuss the formulation of a graph that embeds the floorplan topological information in it. The identification of a valid cut on this graph generates a monotone staircase region in the corresponding floorplan (see Figure 3.2 (a)). The existing methods in [8, 102] employed maxflow based minimum net cut optimization for obtaining an optimal solution. However, these methods have large runtime overhead due to very high time complexity. Moreover, a bipartition with minimum net cut does not always yield a monotone staircase. The work in [103] used

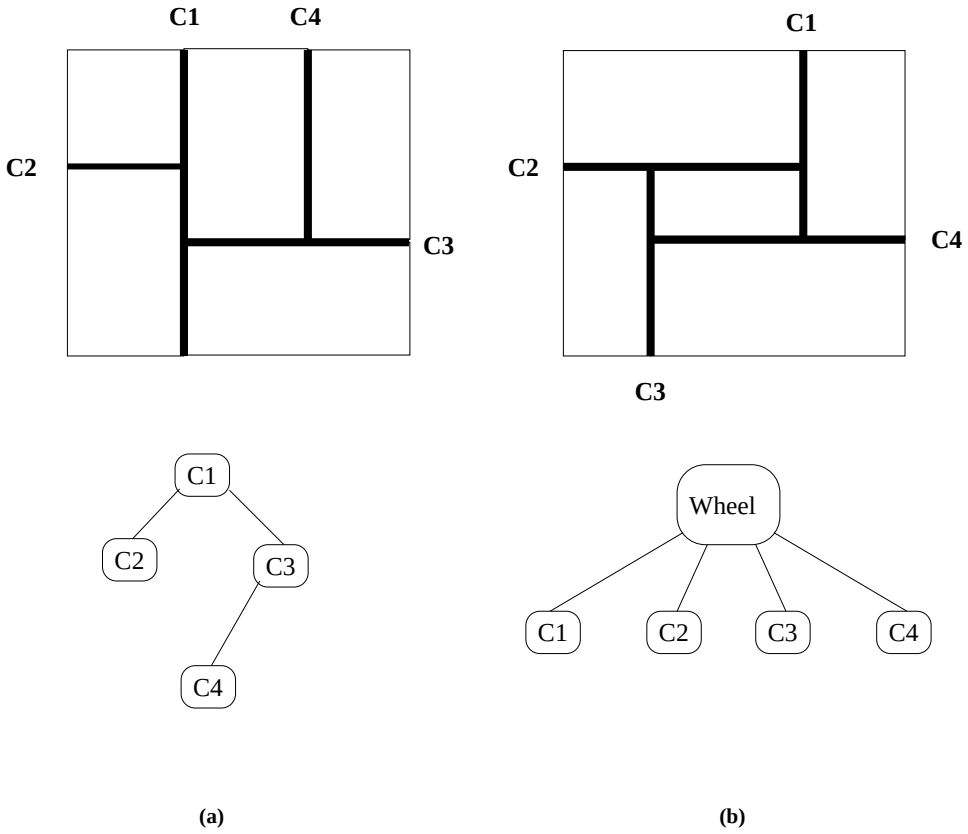
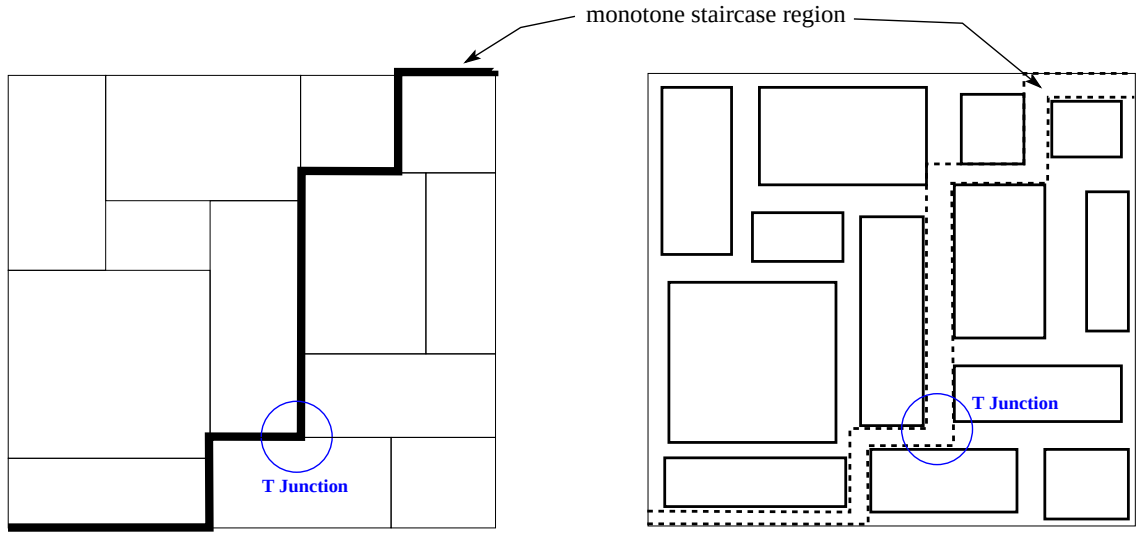


Figure 3.1: Floorplan bipartitioning: (a) slicing tree method, and (b) wheel cut

a faster depth first search (DFS) on a floorplan topology graph in order to obtain an optimal monotone staircase recursively, but without considering any net cut. This method was also restricted to balancing the number of blocks on either side of the partition, but not capable to adapt to area balance.

3.3 Block Adjacency Graph (BAG)

We discuss the formulation of the floorplan topology graph, also called *block adjacency graph* (BAG) [102]. For a design $\mathcal{D}$ having set of blocks $B = \{b_i\}$, set of nets $N = \{n_k\}$ and a given floorplan instance $F(B, N)$ of this design, the unweighted directed graph, BAG $G_b = (V_b, E_b)$ is defined as follows:
$V_b = \{v_i | v_i$ corresponds to block $b_i\}$, and
$E_b = \{e_{ij}\}$, where $e_{ij} = \{(v_i, v_j)$ | block b_i is on the *left of (above)* its adjacent block b_j in $F\}$.



(a) Floorplan of Blocks **(b) Placement of Blocks**

Figure 3.2: Depiction of a monotone staircase in a layout: (a) in a floorplan, and (b) in a placement of blocks [8]

There exists two special vertices in the BAG, namely the source and the sink, that are chosen as the top-left and the bottom-right corner blocks in the given floorplan F respectively. This definition of the BAG gives rise to a *monotonically increasing staircase* (MIS) (C_I as depicted in Figure 3.3(a)) for a given instance of bipartition.

The above definition of BAG is further augmented in [8, 102] to get another graphical framework using the netlist information N. Apparently, the order of this augmented graph is much higher, i.e., ($O(n+k)$ than $O(n)$ for BAG for a design $\mathcal{D}$ having n blocks and k nets. Following this step, they applied their respective graph bipartitioning algorithm in order to obtain a monotone staircase cut in the floorplan. Unlike this method, we use this graph (BAG) straightaway without any such kind of augmentation for identifying a monotone staircase cut in a given floorplan, using our proposed methods discussed in the subsequent sections. In other words, our work does not take any netlist information into account for obtaining the monotone staircases unlike in [8, 102]. Additionally, any graph traversal method [109] on this BAG based bipartitioning framework takes lesser time and also easily scalable for larger designs, while maxflow based methods will have polynomially larger runtime.

We slightly extend the above definition of BAG in order to obtain a *monotonically decreasing staircase* (MDS) by modifying the definition of the edges as follows:

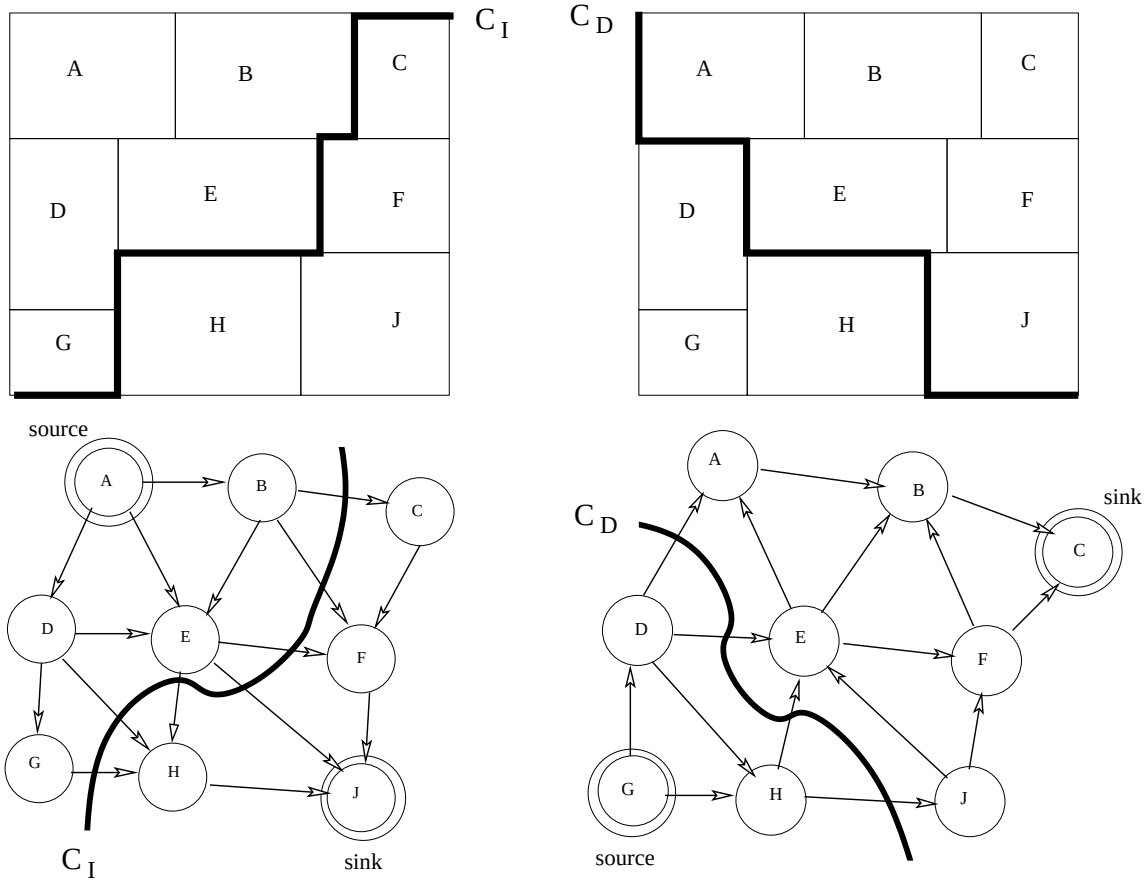


Figure 3.3: A floorplan and the corresponding BAGs for a (a) MIS, and (b) MDS cut

$E_b = \{e_{ij}\}$, where $e_{ij} = \{(v_i, v_j) \mid b_i$ is to the *left of (below)* b_j in $F\}$.

Unlike the earlier definition of the BAG, the source and the sink vertices in this graph are defined as those vertices corresponding to the bottom-left and the top-right corner blocks in the floorplan F respectively. This scenario is captured in Figure 3.3 (b) for an MDS cut C_D. In the rest of the this chapter, we use *ms-cut* as a generic term in order to refer both MIS or MDS cuts, unless stated explicitly and any discussion on monotone staircases equally applies to both of them. Subsequent sections in this chapter will showcase the application of both of these cuts.

Notably, a similar definition for both MIS and MDS variants of a monotone staircase cut on a BAG was seen in [110] as *rising* and *falling* staircase patterns used to define empty non-overlapping (disjoint) polygonal staircase routing regions in a

rectangular layout (floor), but with largest area. This implies that these regions are intended to carry maximal number of nets without any routing obstacles, while our objective is defined for minimizing the number of nets cut a staircase for distributing the nets over the entire bipartition hierarchy. It is evident from Figure 3.4 that not all the cuts on a BAG are monotone staircase cuts. In order to ensure that a cut is certainly a monotone staircase, we consider the following lemma given in [102], known as *monotone staircase property.*

Lemma 1 *If $e_{ij} \in E_b$ is an arc in G_b, then there exists at least one monotone staircase in the floorplan such that the blocks b_i and b_j appear in the left and right partitions respectively, and there exists no staircase with b_i in the right partition and b_j in the left partition.*

Proof Proof of this Lemma is given in [102]. □

In Figure 3.4, we illustrate the working of Lemma 1 for a MIS cut. It shows that all the cut edges in the BAG are forward edges, i.e., directed from the left partition L containing the source vertex towards the right partition R containing the sink vertex and therefore results in a monotone staircase cut. However, in Figure 3.4 (b), the highlighted edge $B \rightarrow E$ in the BAG is directed from R to L, and thus violates the lemma. As a result, we obtain a non-monotone staircase cut and correspondingly a non-monotone staircase routing region in the floorplan. From this illustration and Lemma 1, we observe that it requires at least one back edge directed from R to L in order to yield a non-monotone staircase cut and propose the following corollary.

Corollary 1 *Given the BAG corresponding to a floorplan topology, any cut which has at least one back edge yields a non-monotone staircase cut.*

Proof From Lemma 1, we understand that there can be one or more such back edges necessary for obtaining a non-monotone staircase cut on the BAG. The illustration in Figure 3.4(b) also proves that we need only one back edge for creating a scenario of yielding a non-monotone staircase cut. □

Next, let us address the disadvantage of a non-monotonic routing region over a monotonic one as shown in Figure 3.5. This illustrates three different routing instances of a net $n = (A, B)$ (i) one using a monotonic routing path and (ii) two using a non-monotonic routing path. For the sake of simplicity, we assume that the

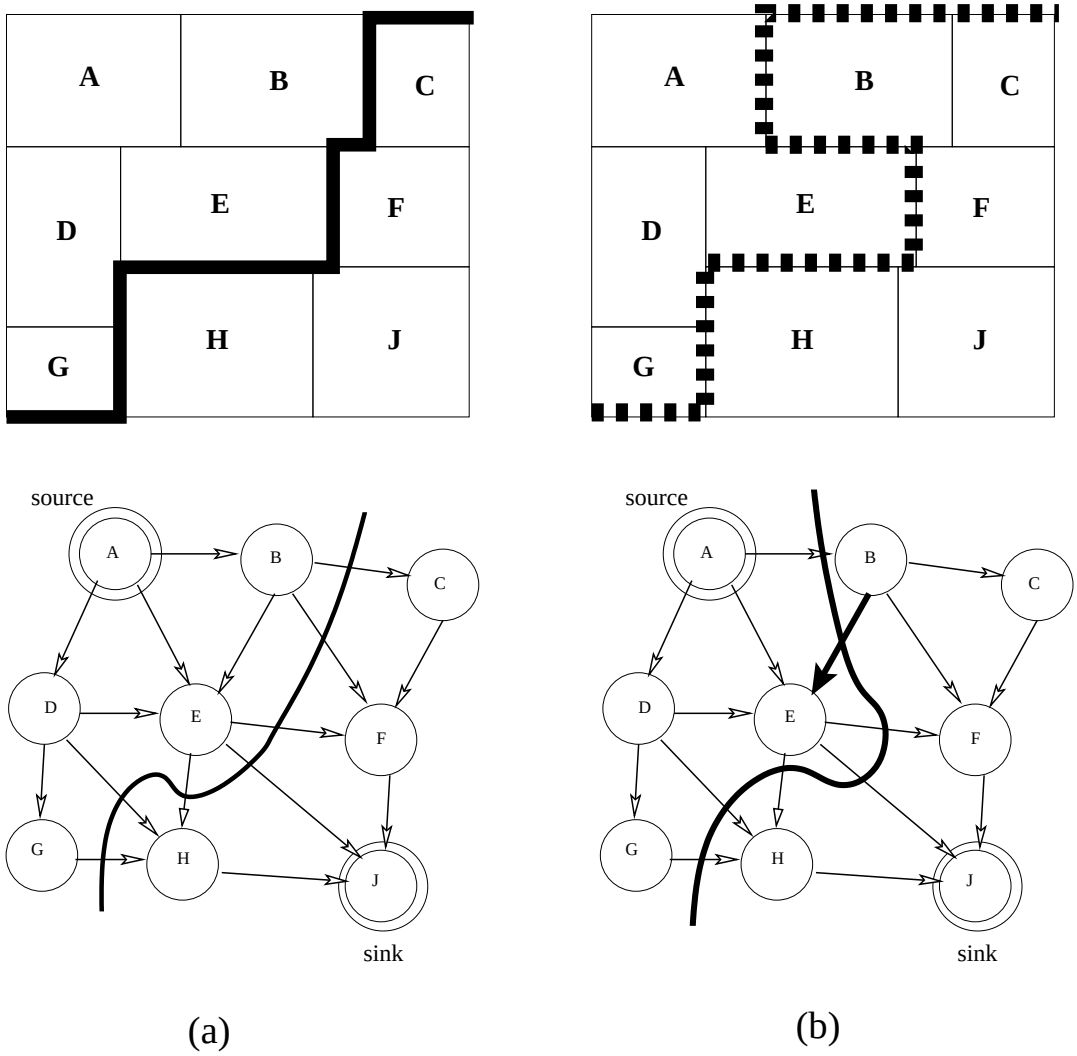


Figure 3.4: Illustration of Lemma 1: for a floorplan with a (a) monotone staircase, and (b) non-monotone staircase

first two routing paths are contained within the bounding box of the net pins A and B; also assume that these routing instances use only two metal layers for routing in the preferred routing direction. This example shows that the first non-monotonic path gives wirelength (in fact, twice the extra-length) more than HPWL of the bounding box compared to the monotonic path which always results in an wirelength equal to HPWL. Another instance of a non-monotonic path yields a detoured routing beyond the net bounding box, resulting in a wirelength more than HPWL.

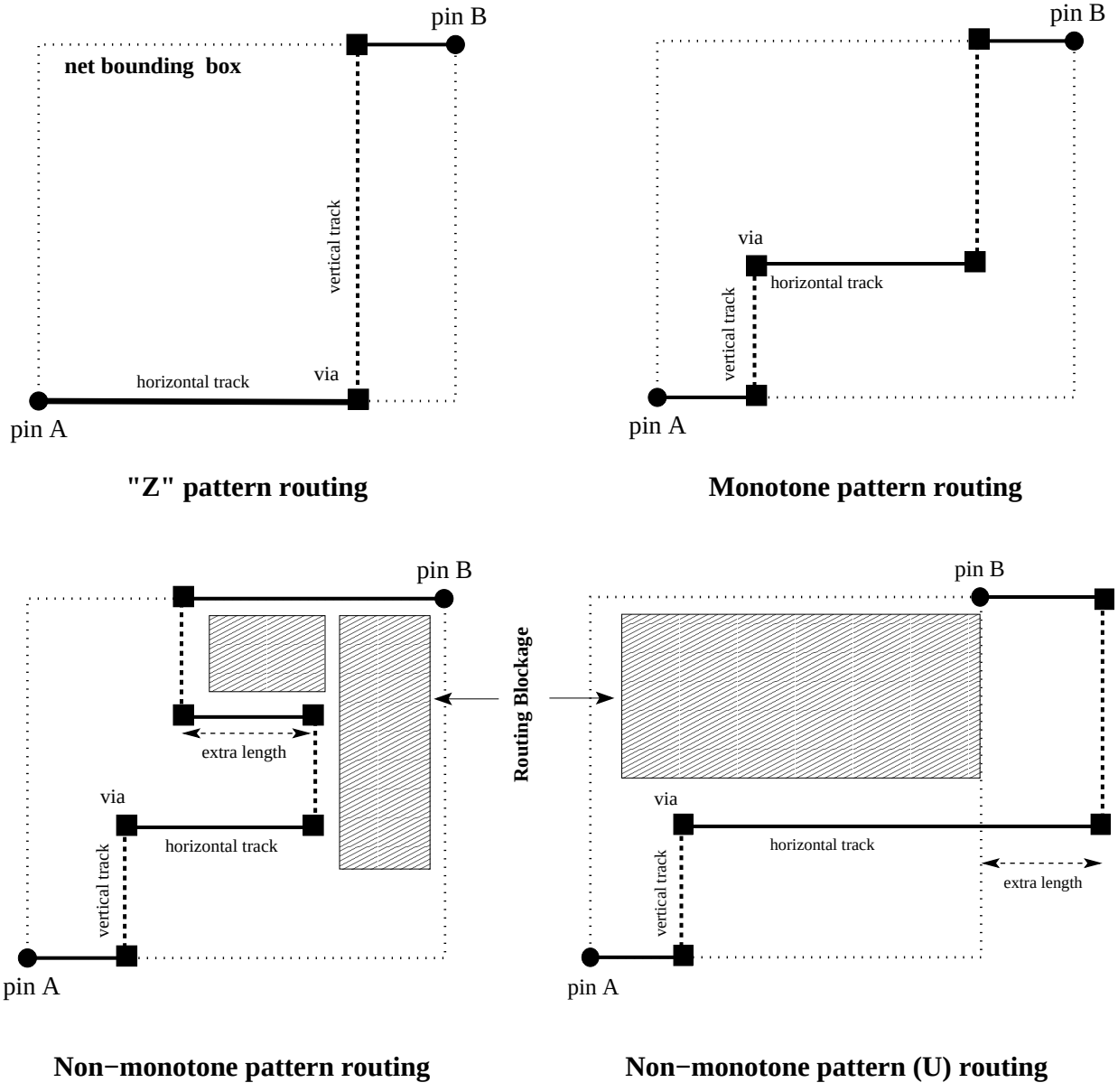


Figure 3.5: Disadvantage of non-monotone staircase routing paths over monotone staircase paths

3.4 A Faster BFS based Greedy Method

In this section, we propose a faster floorplan bipartitioning method entirely based on the topological information of a floorplan instance. This bipartitioning framework identifies a set of monotone staircases by employing breadth first search (BFS) on the floorplan topology graph [102] for a given floorplan without compromising on the quality of the solutions as compared to those obtained by the maxflow based methods

[8, 102]. This method works in a fashion similar to wave-front propagation technique. During each pass of the proposed bipartitioning method, a distinct monotone staircase is obtained while obeying an well defined property on monotone staircases presented in [102] (also see Lemma 1). As discussed earlier in this chapter, such a routing region can either be of increasing or decreasing type, i.e., MIS or MDS depending on user defined configurations (see later in this section). This model is generalized for t-terminal nets ($t \geq 2$) unlike the existing bipartitioners [8, 102]. Next, we propose a variant of this method using depth first search (DFS) on the same floorplan graph. Our aim here is to present a comparative study between both these methods (BFS and DFS) based on the experimental results, since there is no specific reason which method is more apt in our early global routing framework, and entirely depends on the floorplan topology.

In order to obtain a set of monotone staircases for the entire floorplan, we adopt a recursive framework employing this bipartitioning method, presented later in this section and obtain a floorplan bipartition hierarchy (tree), called as *MSC tree* as depicted in Figure 3.1(a). Trivially, there exist a few monotone staircases at the leaf level of this bipartition hierarchy having two blocks as their left and right children. The orientation of these type of cuts is either vertical or horizontal. They are termed as *degenerate monotone staircases* and generally belong to either belong to either an MIS or MDS cut depending on the definition used for constructing the corresponding BAG. This case is a generalized instance of the results obtained by the floorplan slicing method used for *sliceable* floorplans using isothetic (vertical/horizontal) cuts.

3.4.1 Problem Definition

A floorplan bipartitioning problem can be seen as a *set partitioning problem* as discussed in [8] and is shown to be NP-Hard. This problem resembles with an optimal bipartitioning of a finite set $S = \{a_i\}$, where each element a_i has weight(a_i), into (S_1,$S \setminus S_1$) such that the sum of the weights of all the elements in S_1 is equal to that in $S \setminus S_1$. Similarly, a floorplan bipartitioning method aims to obtain an optimal monotone staircase cut on the corresponding BAG such the total area of all the blocks or the number of blocks on one side of the partition (say the left partition L) is equal to that on the other side of the partition (say the right partition R). There exists a couple of variants of this problem and are respectively termed as *area balanced* or *number balanced* floorplan bipartitioning. In other words, we can mathematically

state it as: $P^{opt}(F) = (L^{opt}, R^{opt})$, where $P^{opt}()$, L^{opt} and R^{opt} denote optimum bipartition of a floorplan F, and the corresponding optimum left and right partitions, respectively.

In this section, we present a new framework for both area and number balanced floorplan bipartitioning methods. For a given bipartition solution, some of the nets are cut, while the rest of the nets fall entirely in either of the partitions. Therefore, the floorplan bipartitioning problem is a multi-objective optimization problem [8, 102, 103]. This problem considers both the total area (or the number) of blocks on either side of the partitions and the number of nets cut due to the bipartition; aiming to maximize a balance in the area (or the number of blocks) between each partition and minimum net cut in order to obtain an optimal solution. The optimum solution, which is unlikely to be guaranteed to exist due to hard nature of this problem, consists of the perfect balance among each partition (L, R) implying equal area (or the number) of blocks and the minimum net cut (ideally zero). Therefore, it demands heuristic approaches to be adopted in order to obtain a set of solutions. The best one among these solutions is considered to be an optimal solution.

We summarize the area balanced problem P_{ac} with the following objectives [8, 102]:

1. balance ratio $balr = \min(A_l,A_r)/\max(A_l,A_r)$ be maximized, and
2. number of cut nets(k_c) in the bipartition be minimized,

where $A_{l(r)} = \sum_{b_i \in B_{l(r)}} \text{Area}(\text{b}_\text{i})$, the area of the left (right) partition and $B_{l(r)}$ denotes the set of blocks in the left (right) partition. The number balanced bipartition problem (P_{nc}) [8], can be seen as a restricted version of P_{ac} when the area of each block is almost equal. This implies negligible variance in the area of all the blocks such that they can be normalized to unity. In that case, $balr$ is defined as $\min(n_l,n_r)/\max(n_l,n_r)$, where n_l and n_r denote the respective number of blocks in the left and the right partition.

Since this is a multi-objective optimization problem, we consider a scalar convex function similar to [8] for obtaining an optimal solution as follows:

$$Gain = \gamma.balr + (1-\gamma)(1-k_c/k) \tag{3.1}$$

where k_c is the number of cut nets in a given partition and k is the total number of nets that exist in the given (sub) floorplan and $\gamma \in [0,1]$ is the parameter used to trade-off among the said objectives. In order to obtain an optimal monotone staircase cut in a given floorplan [8], we aim to maximize $Gain$ value such that the area (number)

balance ratio *balr* is maximized while k_c is minimized for a given value of γ. We also need to ensure that the value of this gain is maximum for a suitable choice of γ. The experimental results in [8], suggested γ values to be in the range of [0.4,0.6] for an optimal area/number balanced bipartition.

3.4.2 Illustration: A Sequence of Monotone Staircases

In this section, we propose a new floorplan bipartitioning method using the BAG G_b for a given floorplan F containing a set of blocks B and set of nets N. To make our method much faster than the earlier maxflow based bipartitioners [8, 102], as discussed earlier, our method does not incorporate the netlist information unlike their augmented BAG graph of much higher order than the original BAG and subsequently apply a maxflow [109] algorithm. Instead, this method employs breadth first traversal (BFS) [109] on the BAG and subsequently identifies the cut nets for that partition in linear time. During each pass of BFS, a unique ms-cut is identified. At the end, a sequence of ms-cuts is obtained which is a small subset of all possible ms-cuts in the given floorplan F.

We take an example of Figure 3.6 to illustrate the proceedings of this bipartitioning method using ms-cuts, at any level of the bipartition hierarchy. In this example, we constructed the BAG G_b for the given floorplan for obtaining a sequence of MISes. This method uses a Queue data structure Q for storing the processed vertices of G_b during each pass of the BFS in a level ordering manner. In this example, Figure 3.6 (a) depicts that this method is initialized by enqueing the source vertex A in Queue Q. At any point during the iterations, the elements in Q denote the blocks that are the possible candidates to be added to the left partition L, while those have already been dequeued from Q belong to L. It is also to be noted that each dequeued block once added to L remains in it. This acts as the loop invariant of this iterative bipartitioning method. There is another set of blocks which are not yet explored and thus remain in the right partition R. At any instance, the (L, R) bipartition gives a valid monotone staircase that obeys the *monotone staircase property* (Lemma 1). This process continues until the sink vertex (corresponding to block J) is explored.

In this example, we start with the primitive staircase $\{A\}$ once A is dequeued from Q and added to the left partition L, completing $Level = 0$ in BFS traversal. In order to process its adjacent vertices $\{D, B, E\}$ in R, A is dequeued from Q and they are explored one by one. At this point, except E, B and D are eligible to be added

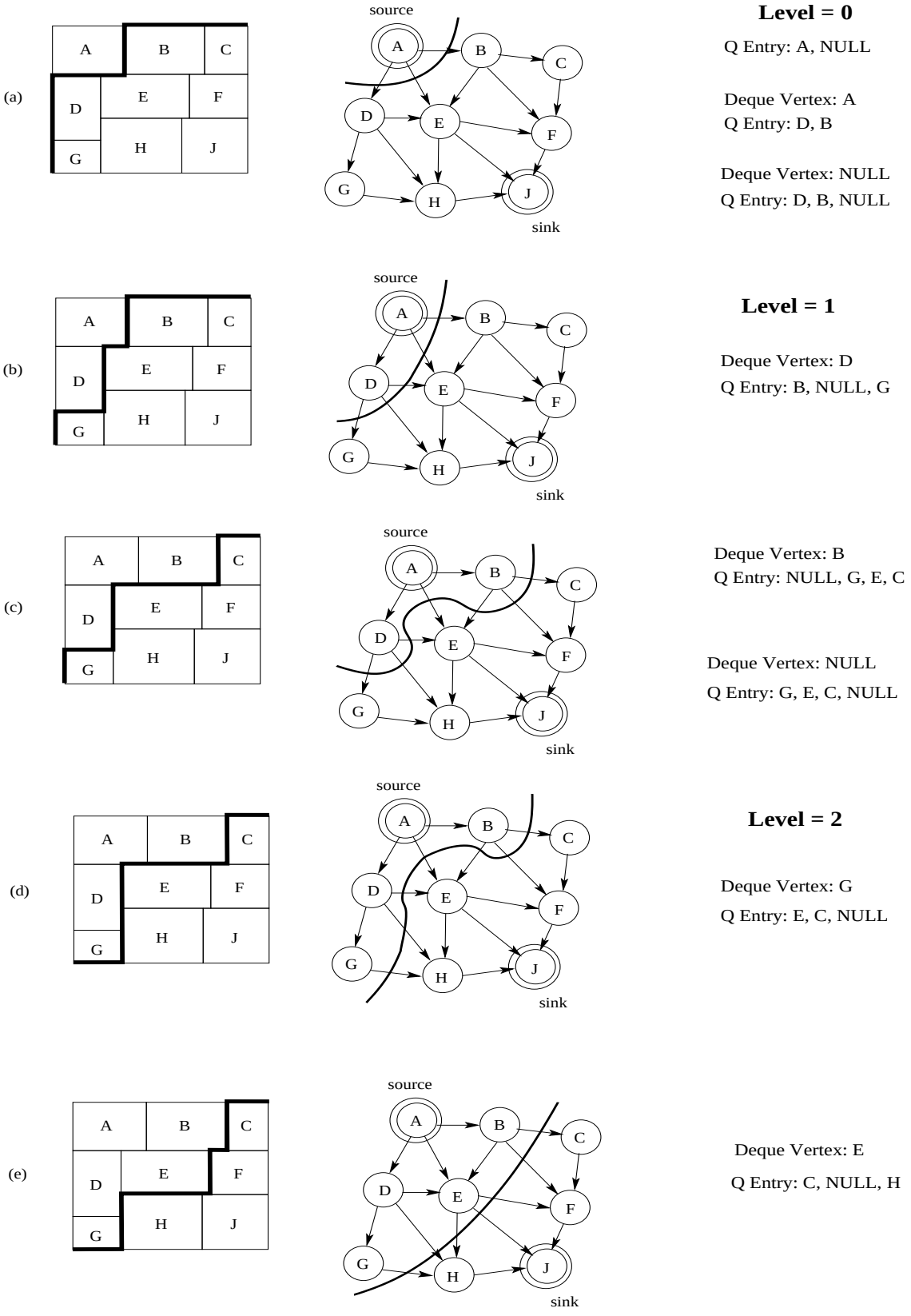


Figure 3.6: Illustrating the working of the BFS based bipartitioning

(in any order) to the left partition. Addition of B and D in Q do not account for any *back edge* from the right to the left partition (Lemma 1). In the current work, we explore the adjacent vertices based on an increasing order of lower left x coordinate values. Thus for the dequeued vertex A, D is given the priority for adding it to Q over B as illustrated in Figure 3.6 (b) and (c), leading to a sequence S of distinct monotone staircases $\{\ \{A\}, \{A, D\}, \{A, B, D\}$. Although the reverse order (selecting B first over D) would also have been considered leading to a different sequence $S' = \{\ \{A\}, \{A, B\}, \{A, B, D\}\}$, we prefer the proposed order of exploring the adjacent vertices. In this case, if E were added to the left partition prior to B and D, there would be two back edges (B, E) and (D, E) which violate Lemma 1. This completes another level ($Level = 1$) of BFS traversal, pushing E to next level.

In the subsequent pass, the adjacent vertices of each vertex (based on the order they were enqueued) residing in Q are explored. For example, G is considered as the next adjacent vertex of D prior to E being another of it; due to the fact that G has smaller lower left x coordinate than E. Hence E is added to the left partition after D as both conform to Lemma 1. Therefore we obtain two more valid monotone staircase $\{A, B, D, G\}$ and $\{A, B, D, G, E\}$ respectively (see Figure 3.6 (d) and (e)) in the sequence S. In general, we obtain a distinct monotone staircase when a block is added to the left partition. Finally, an optimal monotone staircase is identified as the one having maximum $Gain$ (refer to Equation 3.1).

3.4.3 The Algorithm for the BFS based Bipartitioning

Using a specific type of BAG (MIS or MDS), at each level of the bipartition hierarchy, we obtain a sequence of monotone staircase cuts starting from the trivial one (see Lemma 3 in [102]), where the left partition L contains the source vertex of G_b. Subsequent iterations for adding the adjacent blocks take place until a point of convergence is reached, ideally at $W/2$. This is not feasible in case of area balanced bipartitioning, neither in case of number balance mode when the number of blocks is odd. Therefore, an user defined parameter ϵ ($\ll 1$) is used to compute a set of weight bounds $[(1-\epsilon)W/2, (1+\epsilon)W/2]$ [111] such that an optimal bipartition solution (ms-cut) of MIS/MDs type with maximal area balance falling within these bounds. In case of area balanced bipartition, the total weight W is computed by summing the area of all the block in F, whereas in number balanced mode, the same refers to the total number of blocks n in the given floorplan F.

The proposed bipartition method converges when the weight of the left (and hence the right) partition is within the said weight bounds. Notably, it takes longer convergence time with *balr* approaching closer to 1 when ϵ is very small ($\ll 1$) [107]. Conversely, a larger ϵ value leads to a faster convergence but yields an inferior *balr* value leading to a poor sub-optimal solution. In our example presented in Figure 3.6 (e), this procedure is said to converge to give an optimal solution when the left partition L is $\{A, B, D, E, G\}$. However, there may be cases when the convergence within the specified bounds can not be attained and the area of either partition does not fall within the said weight bounds. In that case, we either impose a criteria of maximum number of iterations not to exceed n or wait until the queue is empty, whichever happens earlier.

The nets at a given level of bipartition hierarchy, are either cut nets whose terminals are present on either side of the partition, or belong entirely to either of the two partitions L and R. The nets with all terminals entirely within the left (right) partition are termed as the left (right) *uncut nets*. Furthermore, the cut nets are assigned to the left (right) partition if they have *at least* two terminals in that partition. Thus, in the left (right) partition, the uncut nets along with the (modified) cut nets having terminals belonging to the left (right) partition constitute the total nets for that partition and are to be considered for MIS (MDS) cut generation at the next level of the hierarchy.

The pseudo-code of the proposed bipartitioning method *GenMSCut* is presented in Algorithm 1. We use a parameter named *baltype* in order to represent area (number) balanced bipartitioning, with the respective values of 1 (0) for them and very small values of ϵ. The procedure findPartition() in Algorithm 1 implements this net partitioning method and takes $O(k)$ time.

Lemma 2 *The worst case time complexity for checking the monotone staircase property is $o(n)$, where n is the number of blocks in a floorplan.*

Proof Since the block adjacency graph (BAG) of a given floorplan for n blocks is a planar directed acyclic graph, each vertex in it has a maximum in-degree of p which is much smaller than n, i.e., $o(n)$. Therefore, the time required to check whether the addition of a given vertex to the left partition results in any back edge is maximum p, i.e., $o(n)$. □

Algorithm 1 GenMSCut

Inputs: G_b, N, γ, ϵ, *baltype*

Outputs: An optimal ms-cut with maximal area balance and minimal net cut in the resulting staircase cut with maximum *Gain* value (see Equation 3.1)

Define a Queue Q of size $2n$; S = source(G_b)
Initialize left partition, $L = \varnothing$, $Wt(L) = 0$, $\lambda = \varnothing$
$W = \sum_i Wt(v_i)\ \forall v_i \in V_b$ when $baltype = 1$
$= |V_b|$, otherwise
$lb\_wt = (1-\epsilon)W/2$ and $ub\_wt = (1+\epsilon)W/2$
$level\_no = 0$
Enqueue(S), S.level = $level\_no$
Enqueue($\varnothing$)
while (NOT_EMPTY(Q) $||max\_iteration < n$) **do**
  v_i = Dequeue(Q)
  if ($v_i \neq \varnothing$) **then**
    $L = L \cup \{v_i\}$; Level(v_i) = $level\_no$
    $Wt(L) \leftarrow Wt(L) + Wt(v_i)$
    for ($v_j \in$ adj(v_i)) **do**
      if ($IsValid\_Monotone\_Staircase = TRUE$ and $(Wt(L)+Wt(v_j)) \not> ub\_wt$) **then**
        /* $IsValid\_Monotone\_Staircase$ implies Lemma 1 */
        Enqueue(v_j)
      end if
    end for
    /* here B_l (B_r) corresponds to L ($R = V_b - L$)*/
    findPartition(B_l, B_r, N)
    /* returns N_l (N_r), and N_c being the set of left (right) and cut nets respectively */
    Calculate *Gain* (see Equation 3.1) using $B_l, B_r, k_c = |N_c|, k = |N|$
    $C_p = \langle Gain, B_l, B_r, N_l, N_r, N_c \rangle$
    $\lambda \leftarrow \lambda \bigcup \{C_p\}$
  else
    Increment $level\_no$
    Enqueue($\varnothing$)
  end if
  Increment($max\_iteration$)
end while
Return optimal ms-cut $C_{max} \in \lambda$ with maximum *Gain*

Lemma 3 *The worst case time complexity for generating a monotone staircase cut at any given level of the bipartition hierarchy is $O(n)$, where n is the number of blocks in a floorplan.*

Proof We start from the source vertex of BAG and traverse all the adjacent vertices in a BFS manner and simultaneously check for the validity of the *monotone staircase property* to enqueue them in the queue. Thus it has $O(n)$ enqueue operations in the worst case for all the blocks except the sink vertex, and $o(n)$ operation to check the staircase property for each vertex. Since the BAG is a planar graph, the cardinality of the edge set E_b is $m = O(n)$. The amortized analysis for each vertex yields a total of $O(m)$ time for all enqueued vertices. Thus the total time complexity turns out to be $O(n + m)$, which is again $O(n)$. □

Lemma 4 *The number of monotone staircase cuts obtained by Algorithm 1 in a floorplan having n blocks is $n - 1$.*

Proof Once a block is added to the left partition, it remains in the same partition and results in a distinct monotone staircase. This also holds true when the left partition contains only the source vertex of the BAG. Subsequently, addition of one or more blocks takes at most $n - 1$ iterations until the sink vertex is explored. Thus, the number of distinct monotone staircases obtained by this algorithm corresponds to the number blocks added in each step to the left partition. □

3.4.3.1 The Recursive Framework

The top-down hierarchical procedure presented in Algorithm 2, namely *GenMSC-Tree*, generates a bipartition hierarchy or alternatively called as *MSC tree* for a given floorplan F having a set B of n blocks and a netlist N of k nets. At each node of the hierarchy, *GenMSCut* generates an optimal ms-cut, while *ConstructBAG* constructs the BAG discussed in earlier in this chapter. During BAG construction at each level of hierarchy, the adjacency list of each vertex is sorted with the non-decreasing value of the lower left x coordinates of the corresponding blocks in the floorplan.

Here, *stype* and *baltype* denote the staircase type (MIS/MDS) and the balance type (Area/Number) respectively. The first call to *GenMSCTree* is for the entire floorplan F with $stype = 1$, $Root\_node = TRUE$ for an MIS cut, and specific values of γ and ϵ. The resulting ms-cut (MIS) is the *root cut* of the corresponding bipartition (MSC) tree. The successive recursion of *GenMSCTree* gives rise to the entire

bipartition hierarchy with MIS and MDS cuts at the alternate levels by toggling the value of *stype* between 1 (MIS) and 0 (MDS) respectively.

Algorithm 2 GenMSCTree

Inputs: B, N, F,*stype*,γ, ϵ,*baltype*
Outputs: The *MSC tree*, with increasing (decreasing) ms-cuts at alternate levels

```
if (Root_node || TreeLevel%2 = 0) then
   stype = 1
else
   stype = 0
end if
G_b = ConstructBAG(B, F, stype)
Node.cut = GenMSCut(G_b, N, γ, ϵ, baltype)
Node.Level = TreeLevel; increment TreeLevel
if (|B_l| ≥ 2) then
   Node.left = GenMSCTree(B_l,N_l,F_l,stype,γ,ϵ,baltype)
end if
if (|B_r| ≥ 2) then
   Node.right = GenMSCTree(B_r,N_r,F_r,stype,γ,ϵ,baltype)
end if
return Node
```

An example of MSC tree obtained by the proposed recursive bipartitioning method is illustrated in Figure 3.7 along with the corresponding monotone staircases for the entire floorplanned layout for $n = 17$. This MSC tree has total 16 ms-cuts (MIS/MDS) $(C0 \cdots C15)$ as per Lemma 5.

Lemma 5 *Given a floorplan with n blocks, the MSC tree $T_m = (V_m, E_m)$ corresponding to a set $\mathcal{C}$ of monotone staircases has $n-1$ ms-cuts (internal nodes).*

Proof In a full binary tree, an internal node has two children (out degree = 2); whereas an external (leaf) node has no children (out degree = 0). In our case, the internal nodes correspond to the ms-cuts in the MSC tree $T_m = G(V_m, E_m)$, and the external nodes are the blocks in the given floorplan.
Hence $\sum_i \text{OutDeg}(v_i) = |E_m| = |V_m|$ - 1 = ($|\mathcal{C}|$ + n) - 1
Therefore,
$\Rightarrow 2 \times |\mathcal{C}| + 0 * n = (|\mathcal{C}| + n)$ - 1; here $|\mathcal{C}|$ denote the total number of ms-cuts in the MSC tree.
$\Rightarrow |\mathcal{C}| = n - 1$. $\square$

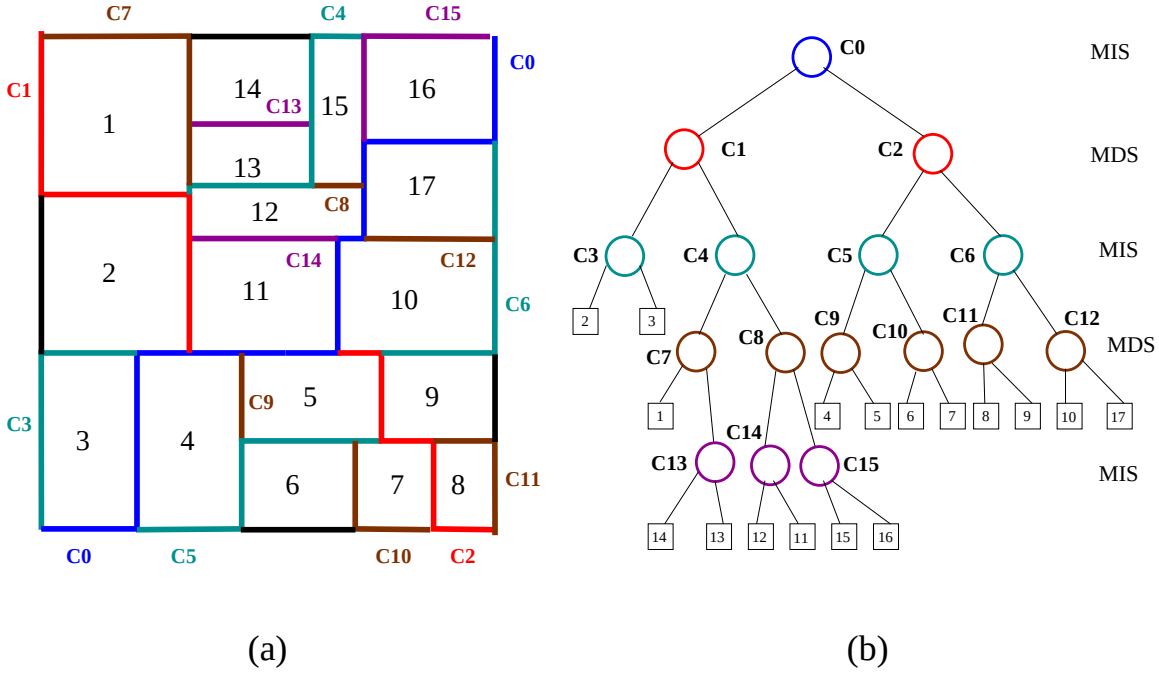


Figure 3.7: The recursive bipartitioning framework: (a) a floorplan with overlaid monotone staircases, and (b) the corresponding bipartition (MSC) tree

Theorem 1 *Algorithm GenMSCTree for hierarchical monotone staircase cut generation takes $O(nk \log n)$ time, where n and k are the respective number of blocks and nets in a floorplan.*

Proof There are at most $n-1$ distinct monotone staircases in the worst case and for each of them, net bipartition takes $O(k)$. Therefore, each hierarchy of the proposed bipartitioning framework requires $O(nk)$ time for identifying an optimal monotone staircase. The procedure *ConstructBAG* takes $O(n)$ since BAG is a planar graph. The depth of the hierarchy, i.e., height of the MSC tree is $O(\log n)$, as the tree is (nearly) a balanced binary tree. Hence, the time required to obtain a hierarchy of optimal monotone staircases is $O((n+nk)\log n)$, i.e $O(nk \log n)$. □

3.4.4 A DFS based Greedy Method

Here we present the pseudo-code, namely *GenMSCut_DFS* in Algorithm 3, where we employ depth first traversal (DFS) on the block adjacency graph (BAG) of a given floorplan for identifying a monotone staircase in it, as compared to the BFS traversal based bipartitioner presented in Algorithm 2. The objectives of this bipartitioner

remain the same as the previous bipartition method using BFS traversal for both area and number balanced mode as: (i) the area (number) of the blocks in each partition to be maximally balanced, and (ii) the number of cut nets due to this bipartition be minimal. In this method, like the BFS based one, we give an weighted emphasis on both the objectives with a trade off parameter $\gamma \in [01]$. In order to measure the quality of a bipartition (hence a monotone staircase), we use the same *Gain* function defined in Equation 3.1. We also use the same recursive framework as in Algorithm 2 for obtaining a hierarchy of monotone staircases for the entire floorplan. Due to the limitations of ϵ on the number of possibly good solutions (monotone staircases) from which the best is chosen as an optimal solution, we omit it in this DFS based algorithmic framework for exploring more solutions (ms-cuts) of floorplan bipartitioning.

Algorithm 3 GenMSCut_DFS

Inputs: G_b, N, γ, *baltype*
Outputs: An optimal ms-cut for a given γ with maximal area balance, and minimal net cut

```
Initialize a Stack S and the left partition L = ∅
Push the source vertex of G_b in S, and include it in L (right partition R = V_b \ L)
while S is not empty do
  Let v_i be the top of Stack S
  if (v_i has at least one adjacent vertex v_j and v_j ∉ L) then
    if (v_i, v_j) results in a valid ms-cut (see Lemma 1) then
      Push the vertex v_j into S and include it in L
      Compute the bipartitioning objective values for the (L,R) partition (see Eqn.
      3.1) and store them in a list λ
    end if
  else
    remove v_i from the top of Stack S
  end if
end while
Return an optimal ms-cut C_max ∈ λ with the maximum Gain value
```

The proposed DFS based bipartitioning method in Algorithm 3 takes the BAG G_b for a given floorplan F and the corresponding net connectivity information N in F as inputs. We also specify a γ value while invoking Algorithm 3 and the type of bipartition, i.e., area or number by assigning a value of 1 or 0 respectively to *baltype*. During the initialization, a stack S is defined and the set of blocks in the left partition

L is set to $NULL$. We start with the source vertex of G_b and push to stack S as well as in L. At this instance, the top of S contains only the source vertex say v_i. We examine the adjacency list of v_i from left to right based on their lower x coordinates and pick one adjacent vertex v_j with the lowest x coordinate. If adding v_j to L does not violate the monotone staircase property (see Lemma 1), v_j is pushed to the top of S as well as added to L. Otherwise, we search for the next adjacent vertex v_k of v_i in the same order of increasing x coordinate and repeat this process until we find an adjacent vertex that does not violate Lemma 1 in order to obtain a new top of S. At any point of time, the top of S represents the lowest leaf in that branch of the resulting DFS tree. We repeat the above procedure of growing the branch till we reach the lowest possible leaf in that DFS tree. Then we start popping the top of S when all of its adjacency vertices has been explored and the DFS tree has grown accordingly following Lemma 1 during a valid top of S is identified. This bipartitioning procedure ends when the source vertex of G_b is popped out of S. At any point of time, when a new top of S is identified, the corresponding parameters are computed (see Equation 3.1).

In Figure 3.8, we illustrate some of the working steps of the proposed DFS based bipartitioning method $GenMSCut\_DFS$ presented in Algorithm 3. In this example, the BAG is constructed for an MIS cut, however can equally hold true for an MDS cut as we stated earlier. At the initialization, the stack S is empty. Then the source vertex A of BAG is pushed to S, also added to L. In the ordered adjacency list $\{D, E, A\}$, D is the first vertex that is eligible to be pushed to the top of S, since adding D to left partition L does not violate Lemma 1. Since D is at the top of S, its adjacency list $\{G, H, E\}$ is explore. Now G qualifies as the first valid vertex to be pushed in S and be added to L. The same procedure continues for the next top of S, i.e., G. In this example, G has only one neighbor vertex H, but does not qualify to be added to L due to the violation of Lemma 1. As a result, G is popped out of S. Subsequent top of S, i.e., D explores its neighbors in the specified order. Like G, none of its unexplored neighbors $\{ H, E\}$ qualifies to be added to the left partition L and hence not pushed into S. Thus, D is also popped out of S.

At this point, the top of S becomes A again, which explores its unexplored adjacency list $\{E, A\}$ resulting in disqualification of E. The last neighbor B is successfully added to L and becomes the current top of S. Then B explores E as its first valid neighbor to be added to L as illustrated in Figure 3.8 (e). This method continues until the sink vertex J of the given BAG is encountered, i.e., all vertices except J

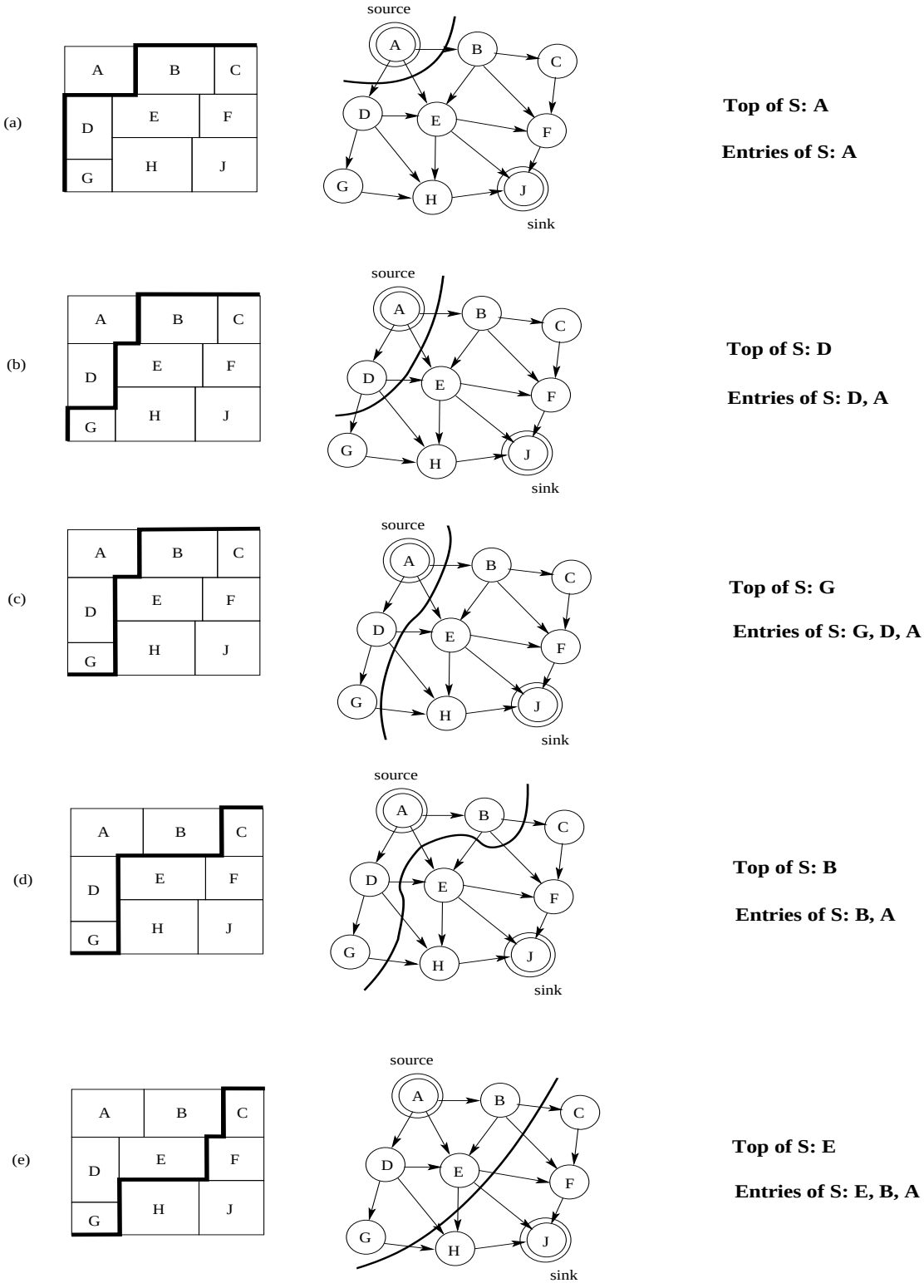


Figure 3.8: Illustrating the working of the DFS based bipartitioning

belong to L. Alike the BFS method, also stated in Algorithm 3, the objective values are computed to obtain the *Gain* value of the corresponding bipartition (ms-cut). It is to be noted that, one vertex added to the left partition L always remain in L and is called the loop invariant.

In order to obtain a balanced bipartition hierarchy of a set of optimal MIS/MDS staircases for the entire floorplan, the proposed DFS based bipartitioning method uses the same recursive framework *GenMSCTree* presented in Algorithm 2, but without the parameter ϵ, for a specified γ value. The height of the corresponding bipartition tree is therefore $O(\log n)$.

3.5 Experimental Results

In order to verify the correctness and efficiency, we tested our algorithms on MCNC and GSRC floorplanning benchmark circuits summarized in Table 3.1. The floorplan instances for each circuit were generated using *Parquet* tool [14] with random seeds. The algorithms presented in this chapter were implemented in C programming language and the experiments were run on a Linux platform (2.8GHz, 4GB RAM).

Table 3.1: MCNC and GSRC Floorplanning Benchmark Circuits [14]

Suite	Circuit	#Blocks	#Nets	Avg. NetDeg
MCNC	apte	9	44	3.500
	hp	11	44	3.545
	xerox	10	183	2.508
	ami33	33	84	4.154
	ami49	49	377	2.337
GSRC	n10	10	54	2.129
	n30	30	147	2.102
	n50	50	320	2.112
	n100	100	576	2.135
	n200	200	1274	2.138
	n300	300	1632	2.161

In the existing maxflow based monotone staircase bipartitioner [8], the authors presented their results for MCNC benchmark circuits only. However, they did not show any results on larger floorplanning benchmarks such as those in GSRC benchmark suite because of much higher time complexity of their method. Their experiments presented several bipartitioning results for different γ values between 0 and

Table 3.2: Comparing the floorplan bipartitioning results obtained by our Faster Method (Algorithm 2) and Maxflow based method [8] for $\gamma = 0.4$ and $\epsilon = 0.05$

Benchmark		Area-balanced mode				Number-balanced mode			
Suite	Circuit	*Gain*		runtime(s)		*Gain*		runtime(s)	
		([8])	(Ours)	([8])	(Ours)	([8])	(Ours)	([8])	(Ours)
MCNC	apte	0.469	0.995	0.013	0.005	0.641	0.920	0.007	0.003
	hp	0.486	0.936	0.016	0.003	0.775	0.933	0.009	0.002
	xerox	0.767	0.989	0.022	0.009	0.723	0.867	0.015	0.007
	ami33	0.752	0.980	0.029	0.014	0.732	0.950	0.021	0.011
	ami49	0.802	0.997	0.095	0.023	0.960	0.954	0.069	0.016
GSRC	n10	-	0.998	-	0.008	-	0.867	-	0.004
	n30	-	0.998	-	0.006	-	0.950	-	0.006
	n50	-	0.987	-	0.050	-	0.950	-	0.011
	n100	-	0.990	-	0.062	-	0.960	-	0.061
	n200	-	0.989	-	0.432	-	0.962	-	0.437
	n300	-	0.979	-	0.656	-	0.962	-	0.617

Note: '-' means no result is available

1. Based on these experiments, they suggested that γ equal to 0.4 is the best choice for an optimal trade-off between the said objectives of the bipartitioning problem. In the area/number balanced optimization problem, a γ value equal to 0 puts the maximum emphasis on minimum net cut, while 1 indicates that our proposed bipartitioner maximum area (number) balance on either side of the bipartition. Any value of γ between 0 and 1 implies their respective weight on those objectives as per Equation 3.1.

In order to measure the effectiveness of our bipartitioner presented in Algorithm 1, the experiments were conducted based on the above suggestion of γ value of 0.4 for both area and number balanced mode. The corresponding results along with those of [8] are presented in Table 3.2 for comparison purpose. In this table, we compare the maximum *Gain* (refer to Equation 3.1) values and runtime for each circuit obtained by our method and [8], for both area and number balanced mode. For most of the circuits in MCNC benchmark suite, our *Gain* values show significant improvement over those in [8]. The runtime obtained for our method also establishes that it is much faster than the maxflow based approach in [8]. Due to the nonavailability of their results on GSRC benchmarks, we could not make a comparison for both the area and number balanced modes.

For further experimentation on the proposed bipartitioning method in Algorithm

1, we constructed a tight example from [8] for the proof of hardness of this (area or number balanced floorplan bipartitioning) optimization problem. This example was created from a floorplan topology containing 11 blocks and 44 nets. This floorplan was derived from the *hp* benchmark circuits in MCNC suite with suitable modifications depicted in Figure 3.9 (a). The dimensions of the individual blocks were computed accordingly. We obtained the maximum *Gain* value as 0.976 for the cut highlighted in Figure 3.9 (b) for γ value equal to 0.4 and $\epsilon = 0.05$. It took only three iterations to converge for the said cut with a runtime of 4 milliseconds. The resulting monotone staircase is highlighted with bold rectilinear segments in Figure 3.9 (a).

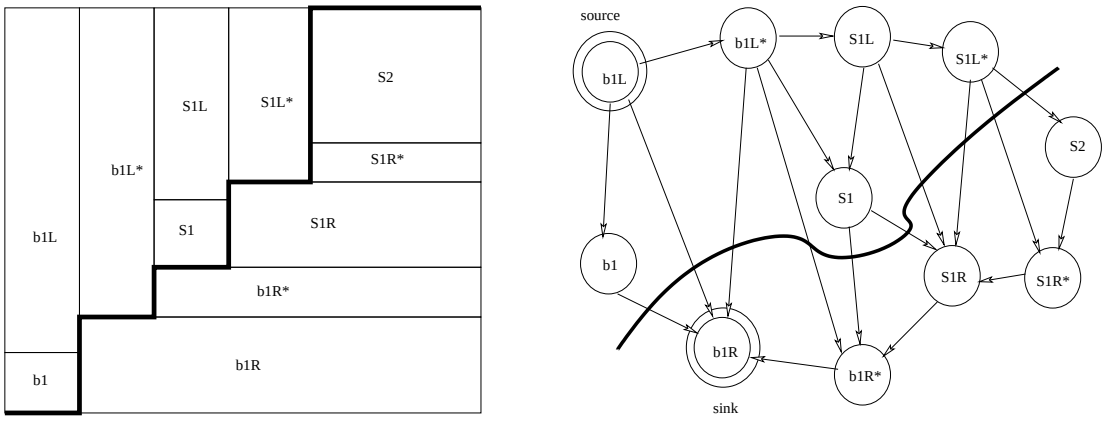
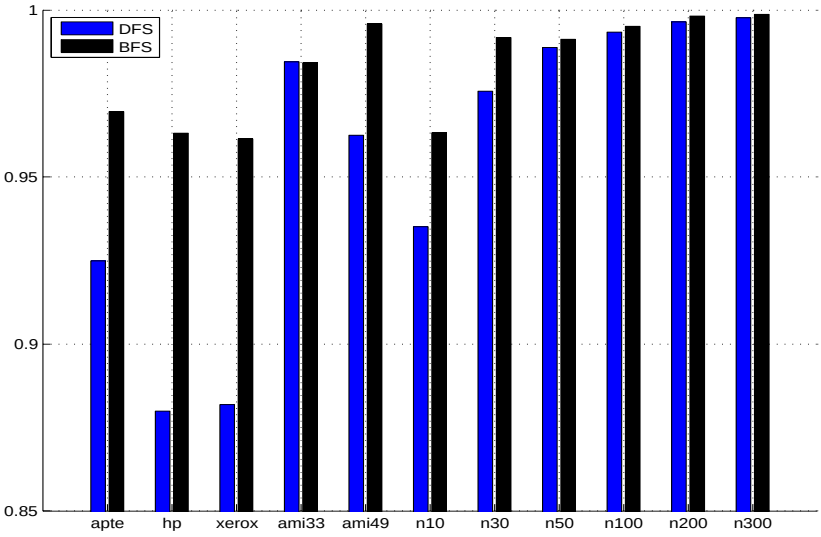


Figure 3.9: Example of a (a) floorplan [8] with an MIS, and (b) the corresponding ms-cut

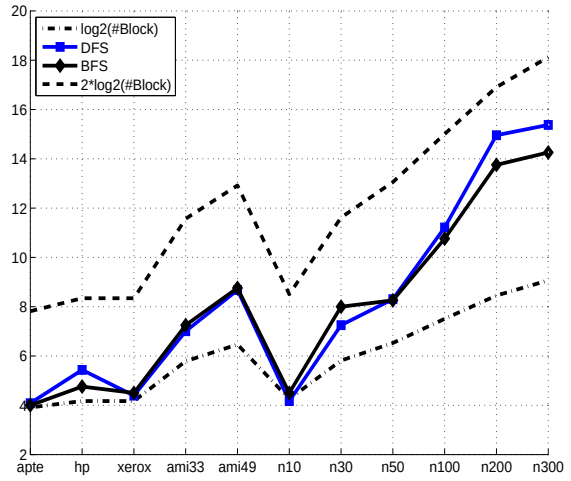
Next, we compare the maximum *Gain* value (refer to Equation 3.1), the average height of the bipartition (MSC) tree, and the runtime obtained for both the proposed BFS and DFS based floorplan bipartitioning methods, namely *GenMSCut* and *GenMSCut_DFS*, presented in Algorithm 1 and 3 respectively. The corresponding comparisons are presented in Figure 3.10. In this experimental setup, we considered γ as 0.4 for both the BFS and DFS methods and ϵ as 0.05 for BFS mode only due to the input requirement in Algorithm 1. From these plots, we see that the BFS method gives better *Gain* values as compared to DFS method for most of the circuits, while runtime is better in case of the DFS method. The height of MSC tree for each circuit is comparable and stays within the bounds of $\log n$ and $2 \log n$, where n is the number of blocks in the floorplan. This implies that the bipartition hierarchy (MSC tree) is a balanced with $O(\log n)$ height as stated in Theorem 1.

Table 3.3: Comparing *Gain* values against γ variation for BFS and DFS methods

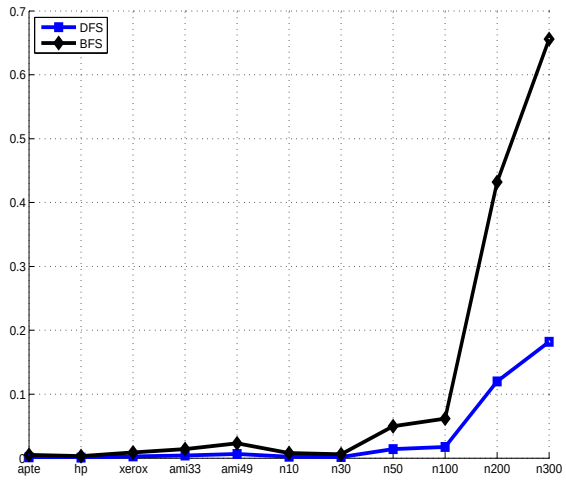
γ	0.1		0.2		0.3		0.4		0.5		0.6		0.7	
Mode /Circuit	**BFS**	**DFS**	**BFS**	**DFS**	**BFS**	**DFS**	**BFS**	**DFS**	**BFS**	**DFS**	**BFS**	**DFS**	**BFS**	**DFS**
apte	0.992	0.993	0.985	0.985	0.977	0.980	0.970	0.970	0.916	0.963	0.954	0.955	0.947	0.948
hp	0.991	0.987	0.982	0.974	0.972	0.972	0.963	0.948	0.855	0.935	0.945	0.922	0.935	0.909
xerox	0.990	0.986	0.981	0.972	0.971	0.963	0.961	0.943	0.954	0.929	0.942	0.915	0.932	0.901
ami33	0.996	0.999	0.992	0.997	0.988	0.995	0.984	0.995	0.853	0.993	0.976	0.992	0.972	0.991
ami49	0.999	0.998	0.998	0.997	0.997	0.997	0.996	0.994	0.843	0.992	0.994	0.991	0.993	0.989
n10	0.991	0.989	0.982	0.977	0.972	0.965	0.963	0.955	0.890	0.943	0.945	0.932	0.936	0.921
n30	0.998	0.998	0.996	0.997	0.994	0.998	0.992	0.994	0.883	0.992	0.988	0.991	0.986	0.989
n50	0.998	0.998	0.996	0.995	0.993	0.993	0.991	0.991	0.871	0.988	0.987	0.986	0.985	0.983
n100	0.999	1.000	0.998	0.999	0.996	0.998	0.995	0.998	0.861	0.998	0.993	0.997	0.992	0.997
n200	1.000	0.999	0.999	0.998	0.999	0.997	0.998	0.996	0.867	0.995	0.997	0.994	0.997	0.993
n300	1.000	0.999	0.999	0.999	0.999	0.998	0.999	0.997	0.881	0.997	0.998	0.996	0.998	0.996
Geometric Mean	**0.996**	0.995	**0.991**	0.990	0.987	0.987	**0.983**	0.980	0.879	**0.975**	**0.974**	0.970	**0.970**	0.964



(a) Max *Gain*



(b) Height of MSC tree



(c) Runtime (sec)

Figure 3.10: Comparing the bipartitioning results of BFS and DFS methods for $\gamma = 0.4$

If we consider *Gain* (see Equation 3.1) as the sole parameter for identifying an optimal monotone staircase, with maximal area balance and minimal net cut, then BFS mode gives better bipartitioning results. Additionally, we present the variation of *Gain* values for each circuit for a set of γ values in Table 3.3. The study of these results shows that BFS yields better *Gain* values for most of the γ values, including 0.4 as prescribed in the maxflow based method [8]. In Chapter 5, we study the variations of the said objective values as well as *Gain* value for each benchmark circuit for different trade-off parameter values.

3.6 Chapter Summary

In this chapter, we proposed a new faster method for recursive floorplan bipartitioning using breadth first search (BFS) method in order to obtain monotone staircase cuts on the floorplan topology graph BAG, over the previously known maxflow based methods such as [8]. Unlike these maxflow based methods, this framework can handle the nets with $t(\geq 2)$ terminals in an unified manner during net partition. Subsequently, we present another bipartitioning method employing depth first search (DFS) on BAG, and subsequently conduct a comparative study with the BFS method. A part of this chapter has been presented in [111].

In this bipartitioning method, our primary aim was to use a simple graph based on the floorplan topological information of the blocks, unlike the maxflow based methods in which the floorplan graph is further extended to another graph using the net connectivity information. Notably, the order of this augmented floorplan graph grows with the increasing number of nets across the circuits (designs). Therefore, this graph model significantly contributes to the time complexity of these maxflow based methods. Moreover, these maxflow based algorithms such as push-relabel algorithm takes $O(n^3)$ time, where n contains both the number blocks and nets in it. In summary, the earlier bipartitioning methods are very time consuming than ours and also not effective in terms of *Gain* values as per the experimental results.

Alike the previous maxflow based works, we considered only two objectives for this multi-objective optimization problem of area (number) balanced floorplan bipartitioning as: (i) the balance in area of each partition, and (ii) the number of cut nets for that partition. Since the area (number) balanced floorplan bipartitioning is a NP-hard problem [8], we greedily pick one solution as an optimal monotone staircase cut from a set of solutions based on a scalar parameter called *Gain* (see Equation 3.1 and [8, 102]). By computing and evaluating this parameter for each of the resulting bipartitions, we identify the best solution as an optimal solution satisfying both the objectives for a given γ value. The experimental results show that BFS based method gives better *Gain* values than the DFS based method, in addition to showing improved *Gain* and runtime over the results reported by the recent maxflow based bipartitioning method [8].

Chapter 4

STAIRoute: The Early Global Routing Framework

4.1 Introduction

In VLSI physical design flow, *global routing* (GR) is an indispensable step towards a successful design closure, with heavy dependence on the floorplanning/placement solution. It acts as an aide to the subsequent *detailed routing* (DR) of the wires through different metal layers. Due to shrinking feature dimensions with technological advances in IC fabrication process, the challenges in the physical design phase are increasing continuously. There has been a tremendous increase in the routing constraints arising from not only stringent layout design rules, but also due to random process variations and the limitations due to sub-wavelength optical effects in the lithography process. These lithography effects are commonly known as *design for manufacturability* (DFM) issues which need utmost attention to be paid for design closure for fault free fabrication of the devices. A successful, global or detailed, routing completion usually needs many iterations in order to meet those ever growing constraints and also meet the stringent criteria of the performance factors in the designs. Due to large number of iterations, these constraints are sometimes relaxed by the designers. In the worst case, the entire design process has to be started over from an earlier stage of PD flow (see Figure 1.9 (a) in Chapter 1) such as floorplanning or placement in case the errors due to the constraints were not taken care properly or could not be satisfied.

4.2 Study on Post-Placement Global Routing

As depicted in Figure 1.9 (a), the global routing stage uses the placement information to identify the routing regions and their capacity. This process divides the entire layout into equal size regions called *global routing bins* or tiles (see Figure 4.2). Each routing bin acts as a vertex in the corresponding grid graph model [17]. The size of these bins plays a significant role in the runtime of the routing engine as well as the quality or effectiveness of the global routing solution obtained at any given time. A smaller sized routing bin gives better accuracy in the routing solutions obtained, but takes longer runtime. On the other hand, a larger bin sizes implies a faster solution, but an inferior one. This model facilitates the routing of a net (segment) from the center of one bin that contains one pin of the net to the other, but not between the two pins. The remaining unrouted net segments (from center of the bin to the pin in it) are handled during detailed routing. This problem is most commonly known as *pin access problem*. There is a special case of this problem, known as *intra-bin* routing, when a net (segment) to be routed falls entirely within a routing bin. These type of nets are not routed during the global routing stage and pushed to detailed routing stage instead. A comparative study in Figure 4.1 captures the instances of both pin accessibility and intra-bin routing in the existing post-placement global routing as well as proposed floorplan level pin accessibility driven early global routing presented later in this chapter.

In a layout with fixed outline area, the placement of standard cells, macros and the pad cells determines the effective routing area in the layout. Since, all the nets can not be routed through the residual routing area using two routing layers, many routing layers are allowed. Depending on the maximum number of routing (metal) layers used by each of these cells, macros and pads for their respective internal routing, the minimum effective routing layer above each of those blocks/cells is determined. Moreover, each of these blocks acts as a routing blockage in the layers below those minimum specified routing layers. The regions beyond these blockages are free to accommodate a net (segment) in any routing layer, up to a maximum permissible routing layer. In the grid graph model, the edge capacity is determined based on these blockages in any routing layer. However, this model allows only a specific routing direction (horizontal or vertical) in each layer; for example a vertical (horizontal) edge allows only an effective number of horizontal (vertical) routing tracks.

The multi-terminal nets are decomposed into two terminal segments using either

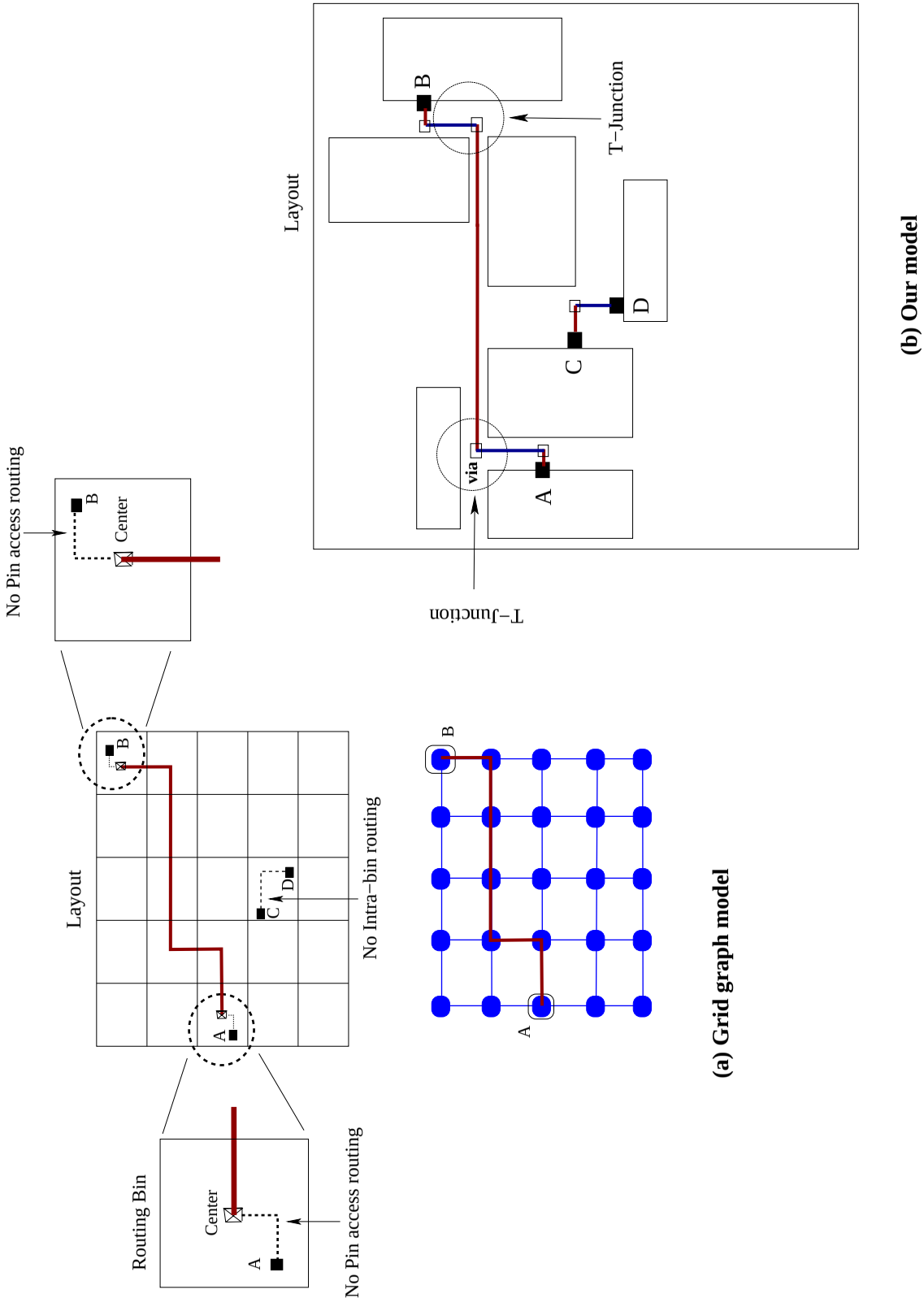


Figure 4.1: Pin access problem during global routing using: (a) grid graph model, and (b) our model

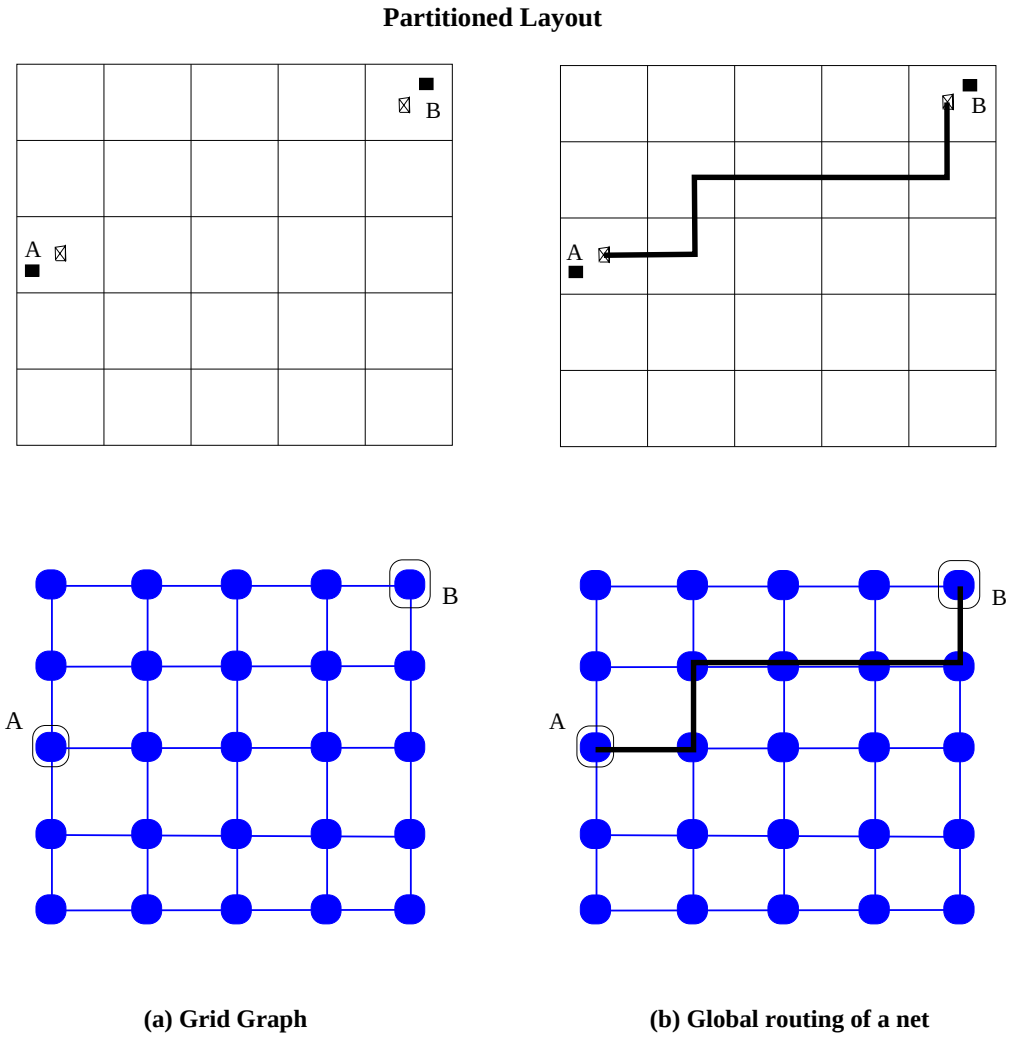


Figure 4.2: Grid graph model and global routing

rectilinear minimum spanning tree (RMST) [22], or *rectilinear steiner minimal tree* (RSMT) [21] topology as the initial routing topology with minimum length, without considering any routing blockage. Subsequently, a congestion driven routing for each two terminal net (segment) is adopted through the routing regions in any of the permissible layers. The congestion models in those methods have been formulated based on the capacity of the grid edges and the routing demands through them, along with a penalty function. Recently, pattern routing [19, 20] has become much popular and effective way of identifying the routing paths. In most of the cases, a planar two layer routing solution is obtained, followed by a congestion driven layer assignment of the nets. Alternatively, a routing method with congestion driven simultaneous layer

assignments has also been proposed in [22]. But this method is more time consuming than the other one. During the layer assignment of the nets, a large number of vias are used to make the inter layer interconnections, without disturbing the two layer routing topology. This is called constrained via minimization (CVM), which is a known to be a hard problem. CVMs are known to be faster than unconstrained via minimization (UVM) methods, but does not ensure any better solution.

Due to unsuccessful routing or over congestion ($\geq$ 100%) in the routing edges, several *ripup and reroute* (RRR) techniques using both pattern and maze routing [17, 33] have been adopted towards the routing completion, occasionally compromising in net length due to detours beyond the bounding box of the respective nets. The major challenge in the event of unattainable routing completion is to get back to placement stage in order to generate a new placement solution, but with no guarantee for successful routing completion. On the contrary, if the failed nets during global routing are pushed to detailed routing stage, the congestion scenario during detailed routing will further degrade due to the routing of the local (intra-bin) nets as well as those subnet routing required for pin accessibility. Nowadays, iterative global and detailed is being done simultaneously when the first global routing solution does not yield 100% routing completion. But, this may lead to several iterations until the goal is achieved. Thus, it may prove to be very costly if is not completed within a stipulated time frame, impacting on time-to-market of the product. However, the possibility of recurring iterations due to uncontrollable failures during the traditional global routing stage may however be reduced if we can devise an early method that predicts a feasible global routing solution for a given floorplan of a design, as highlighted in Figure 1.9 (b). The main idea is to validate the generated floorplan such that the subsequent placement driven by this floorplan topology yields a complete global routing solution of all the nets present in the design. Moreover, this early routing solution may be helpful in guiding the final global routing solution with fewer iterations.

4.3 STAIRoute: The Early Global Router

In this chapter, we present the first of its kind early global routing framework after floorplanning. In this work, we use the monotone staircases as the routing regions through which a set of nets are routed through a set of routing layers. These routing regions are identified by the recursive floorplan bipartitioning methods presented in

Chapter 3. Recently, pattern routing such as single bend (L shaped) [20], two bend (Z shaped) [20, 21], or even more bends such as monotone staircase patterns [19] have gained significant importance in grid graph based global routing methods. With increasing number of bends, these routing patterns yield more flexibility in finding a possible routing path, but at the cost of more vias. Notably, the pattern based global routing is much faster than the maze routing methods as stated in [19, 20]. A trade off between routability and the number of vias has to be made keeping in mind that the routing regions are not heavily congested.

Our congestion model restricts the number of nets through a region up to a maximum of 100% usage of the capacity. The advantage of using the monotone staircases is that these are known to have advantages of *acyclic routing order* for successful routing completion [104, 105] and handle *switch box routing* problem effectively [17]. In case of no fixed outline floorplanning, they allow easy *channel resizability* [104] to mitigate heavy congestion ($\geq$ 100%) in them, specially during detailed routing. Since the routing order of the nets is another critical factor for routing completion, the bipartition hierarchy helps in identifying an well defined routing order.

Outline of the proposed early global router STAIRoute [112] (see Figure 4.3) are as follows:

1. *Identification of the routing regions* as monotone staircases in a given floorplan topology by floorplan bipartitioning methods; a graph theoretic formulation with these staircases is used to determine a feasible routing path for all the nets;

2. *Net ordering* based on half perimeter wire length (HPWL) and the number of terminals (Netdegree);

3. *Decomposing multi-terminal nets* into an equivalent set of two-terminal net segments using minimum spanning tree algorithm and defining a new *Steiner tree* topology;

4. *Routing solution for a given number of metal layers* using a shortest path algorithm to find the best possible routing path while respecting the prevailing congestion scenario (of $<$ 100% utilization) across the layers;

5. *Ensuring no congestion* in the routing regions across a given number of metal layers.

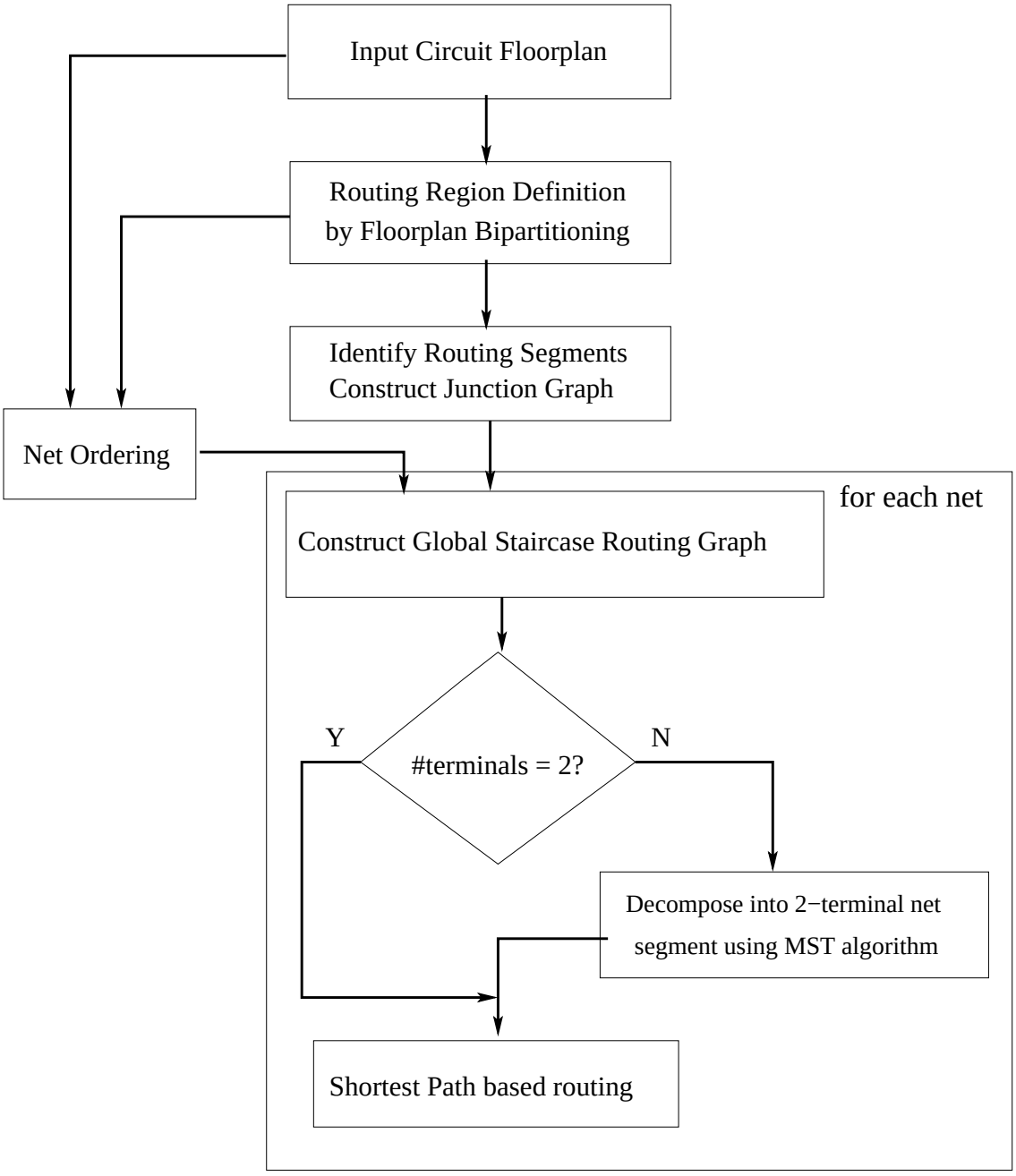


Figure 4.3: Outline of the proposed Early Global Router

4.3.1 The Routing Model

Using a floorplan bipartitioning framework as discussed in the previous chapter, we obtain a set of MIS (MDS) cuts $\mathcal{C} = \{C_i\}$ on a give floorplan F at alternate levels of the bipartition hierarchy (MSC tree). The corresponding monotone staircases

are used as the routing regions for the proposed global routing framework. Each monotone staircase consists of one or more rectilinear (horizontal/vertical) segments bounded by the boundaries of a distinct pair of blocks (see Figure 3.3). For each routing region and its segment(s), the maximum number of nets to be routed through it denotes the *reference capacity rCap* and is obtained from the net cut information in the respective ms-cut nodes in the MSC tree. During routing, the *used capacity* (also known as routing demand) *uCap* of a region gives the information about the number of nets (segments) have already been routed through it. This parameter is initialized to zero before the routing process starts.

4.3.1.1 The Junction Graph

In this section, we present our global routing model based on the monotone staircase routing regions obtained by applying the proposed floorplan bipartitioning methods on a given floorplan. The intersection points between two staircases are called namely T-junctions (see Figure 4.4 (c)). It is evident that there exists a rectilinear segment between each pair of adjacent T-junctions, henceforth referred as *junctions*.

Using the notion of these junctions in a given floorplan, we define an weighted undirected graph (see Figure 4.4(d)), called *junction graph* $G_j = (V_j, E_j)$, where
$V_j = \{J_p$, corresponds to a set of junctions$\}$, and
$E_j = \{\{J_p, J_q\} \mid$ a pair of adjacent junctions $\{J_p, J_q\}$ with a segment s_k of a staircase $C_m \in \mathcal{C}$ between them$\}$.

The resulting junctions of a given floorplan is presented in Figure 4.4 (d). As depicted in Figure 4.4 (c), all the junctions except those near the corners of the floorplan with degree two, have degree of three in G_j, i.e., have three incident edges from its adjacent junction vertices.

Lemma 6 *Given a floorplan F with n blocks, the number of T junctions in it is $2n-2$.*

Proof Each internal face in the BAG G_b of a floorplan F corresponds to a T-junction, and is bounded by three edges. Thus we have $3 \times (f-1) = 2m$ excluding the exterior face, where f and m being the number of faces and edges in G_b respectively. Using *Euler formula* [109] for planar graphs, $n - m + f = 2$, and replacing f by $2m/3 + 1$, we get $m = 3(n-1)$.

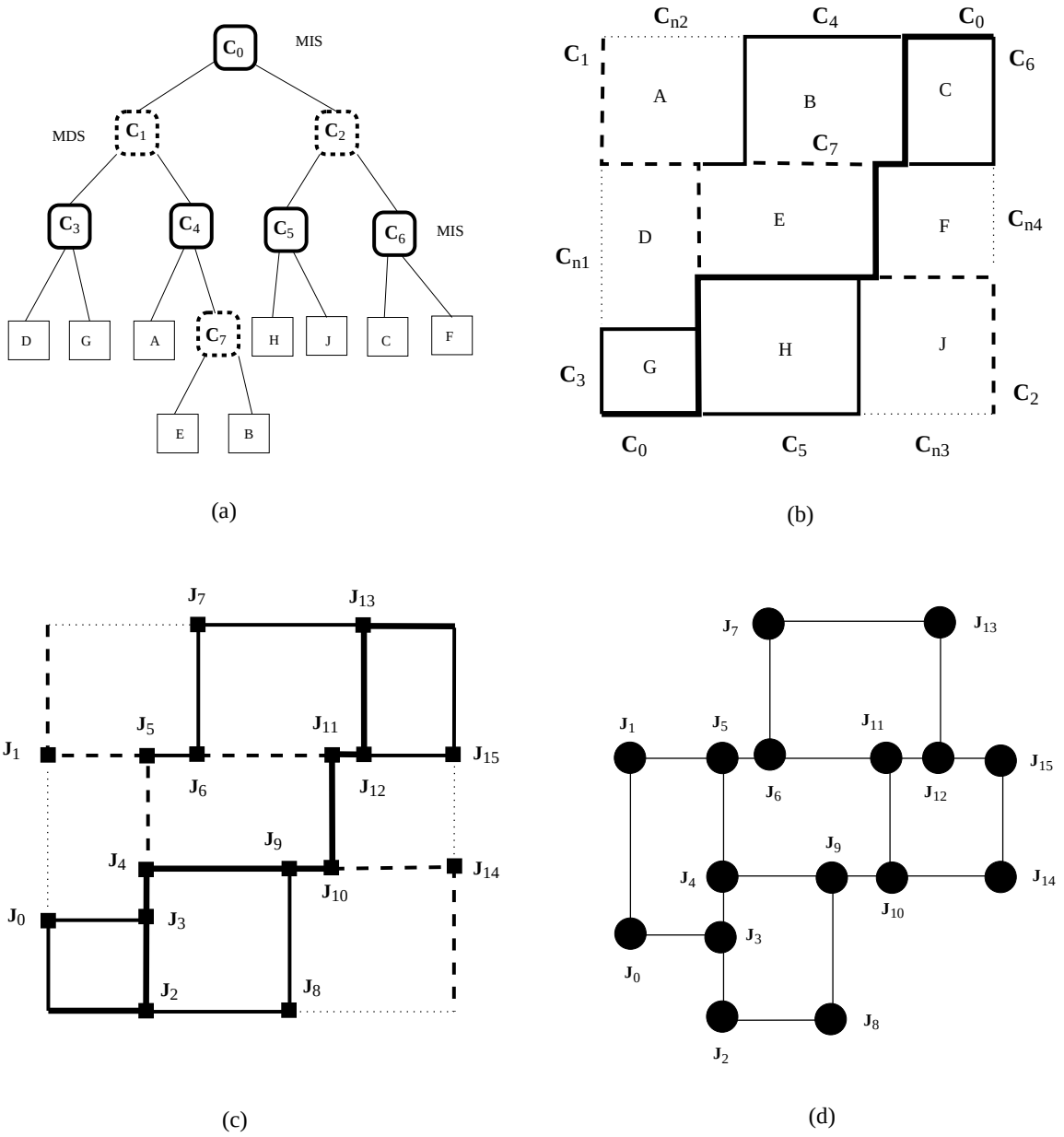


Figure 4.4: A floorplan with: (a) its bipartition hierarchy MSC tree, (b) the corresponding monotone staircases, (c) T-junctions at which these monotone staircases intersect, and (d) the corresponding junction graph.

Hence, the number of T-junctions in F,
$f - 1 = 2m/3 = 2n - 2$ □

Lemma 7 *The construction of the junction graph takes $O(n)$ time, where n is the number of blocks in the floorplan.*

Proof By Lemma 6, we know that there are $O(n)$ edges in the BAG, where each edge corresponds to a rectilinear routing segment in a monotone staircase routing region. Therefore, for each segment s_k having a pair of junctions $\{J_p,J_q\}$ as its endpoints, an edge is inserted in the G_j. Hence, the construction of the junction graph G_j takes $O(n)$ time. □

Observation 1 *Based on Lemma 6 and our study on several floorplan topology, it can be shown that* $|E_j| = 3n - 7$ *holds true for most of the floorplan topologies, such that each boundary of the corresponding floorplanned layout contains at least one junction.*

We illustrate this observation in Figure 4.5. In this example, both the floorplans contain 6 blocks and hence 10 junctions (following Lemma 6). The instance (a) shows that it has at least one junction on all the boundaries of the floorplan, while the instance in (b) shows no junction at the bottom boundary adjoining the block D. The corresponding junction graph for the instance (a) has 11 edges obeying $|E_j| = 3n - 7$, but (b) has 12 edges. Similarly, for the junction graph G_j presented in Figure 4.4 (d), for $n = 9$, we notice that it has 16 vertices and 20 edges as per the observation made above.

4.3.1.2 The Congestion Model

The weight of each edge $e_{pq} \in E_j$ is computed as:

$$wt(e_{pq}) = length(s_k)/(1 - p_{s_k}) \tag{4.1}$$

where p_{s_k} is defined as the *Congestion* in s_k in a metal layer M_i, also known as *normalized usage*:

$$p_{s_k} = uCap_{s_k}/rCap_{s_k} \tag{4.2}$$

The denominator $(1 - p_{s_k})$ in Equation 4.1 is defined as the *usage penalty* or *congestion penalty* on the edge weight for routing a net through s_k in a particular metal layer M_i. In Figure 4.6, we illustrate the variation of edge weight with respect to the *normalized usage* p_{s_k}.

In this routing framework, over congestion is avoided in all the segments in any routing layer by restricting p_{s_k} to 1.0. This implies that no routing region allows the routing demand of s_k to exceed its routing capacity in any routing layer. This

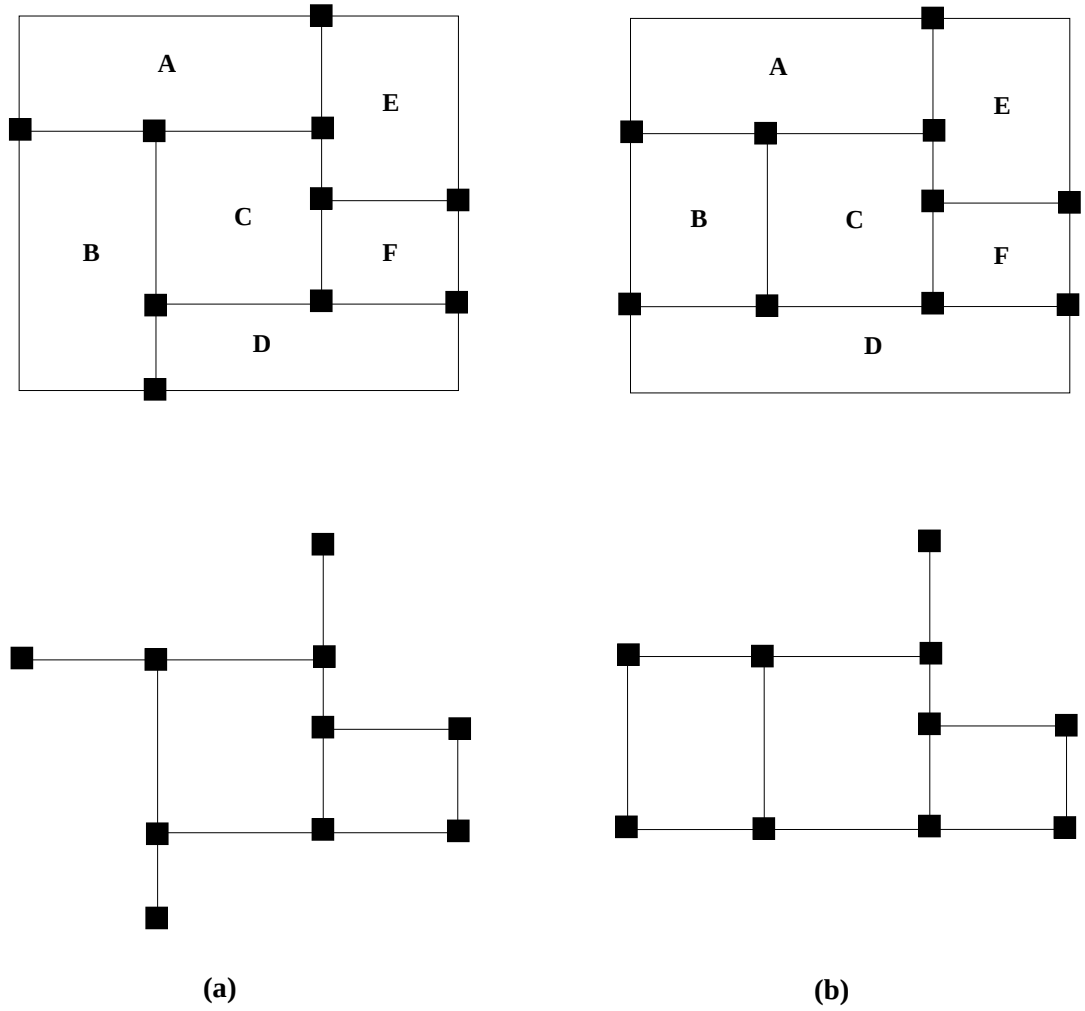


Figure 4.5: A floorplan with T-junctions: (a) at least one junction on each boundary, and (b) no junction at the bottom boundary

is achieved by setting the edge weight to *Infinity* whenever p_{s_k} becomes 1.0, i.e., routing demand is equal to the routing capacity. The corresponding edge in that particular routing layer becomes virtually nonexistent in E_j. This ensures that the over congestion case of $p_{s_k} > 1.0$ does not occur at all. In Figure 4.6, we illustrate the regions marked by $p_{s_k} \leq 1.0$ and $p_{s_k} > 1.0$ as *Under-Congestion* and *Over-Congestion* regions respectively. The proposed congestion model restricts the routing to *Under-Congestion* region only. However, it may be noted that routing may fail for some of the nets due to insufficient capacity in some of the routing regions for a specified number of metal layers.

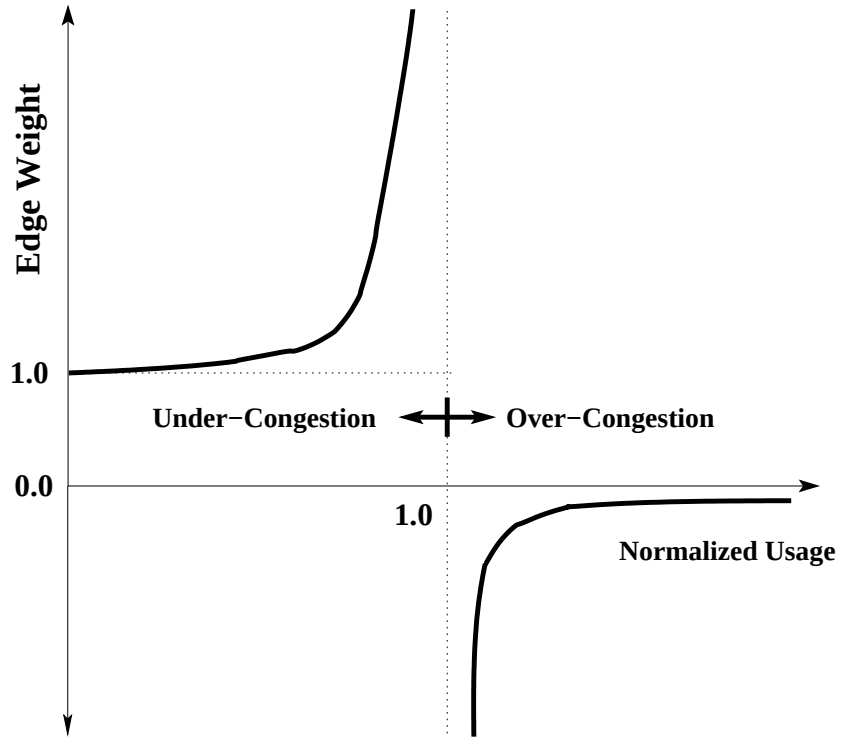


Figure 4.6: Junction Graph Edge weight ($wt(e_{pq})$) vs. normalized usage (p_{s_k})

4.3.1.3 The Global Staircase Routing Graph

In order to realize the proposed early global routing framework, we extend the junction graph G_j for each net $n_i \in N$, where N is the set of nets to be routed. For each t-terminal ($t \geq 2$) net $n_i \in N$, we use $G_j = (V_j,E_j)$ as the backbone graph to derive an augmented version of it. The corresponding augmented graph for net n_i is called *Global Staircase Routing Graph* (GSRG) $G_r^i = (V_r^i,E_r^i)$. The definition of this graph is follows:
$V_r^i = V_j \bigcup \{t_l | t_l \in n_i\}$, and
$E_r^i = E_j \bigcup E_{lp}$
Here, each pin-junction edge $e_{lp} \in E_{lp}$ is defined as:
$e_{lp} = \{t_l,J_p\} \mid \forall t_l \in n_i$ and $\exists J_p \in J$, the pin t_l resides on a segment s_k associated with the junction $J_p\}$.

Alike the edges in junction graph, we calculate the weight of a pin-junction edge e_{lp} as below:

$$wt(e_{lp}) = \text{distance}(t_l, J_p)/(1 - p_{s_k}). \tag{4.3}$$

and define $(1 - p_{s_k})$ as the *usage penalty* on the edge weight for routing a net through

the corresponding segment s_k. An example in Figure 4.7 (a) and (b) illustrates the construction of GSRG G_r^i for (a) 2-terminal, and (b) 3-terminal nets respectively.

Lemma 8 *For any t-terminal net $n_i \in N$, the construction of the corresponding GSRG G_r^i takes $O(t)$ time.*

Proof For a given net n_i with t terminals in it, the corresponding GSRG $G_r^i = (V_r^i, E_r^i)$ is obtained by augmenting the junction graph $G_j = (V_j, E_j)$. In other words, V_j is augmented by the vertices pertaining to all t terminals in n_i in order to obtain the vertex set V_r^i. It is also to be noted that each terminal resides on a segment s_k, having a pair of junctions (J_p, J_q) on either ends. Therefore, each terminal (pin) contributes two pin-junction edges and thus contributes a total $2t$ edges to G_r^i for all t terminals. Hence, the construction of G_r^i takes $O(t)$ time for each net n_i. □

After successful routing of each net n_i through a set of rectilinear segments $\{s_k\}$, the used capacity (routing demand) $uCap_{s_k}$ of each segment is augmented by one. Subsequently, the weights of the edges in G_j and then in G_r^{i+1} are updated before the identification of a routing path starts for the subsequent net n_{i+1}. As cited before, when the congestion in any given segment s_k increases further pushing its normalized usage p_{s_k} to approach the value of 1.0 in a particular routing layer M_i, the weight of the corresponding edge in G_r^i tends to become *Infinity* (see the congestion profile in Figure 4.6). At this instance, this edge is least favorable for routing due to very high routing cost. No further routing is possible through such a segment with 100% normalized usage in M_i and the relevant edge in G_j (and hence in G_r^i) for that layer virtually disappear (by making edge weight Infinity). If there is another higher permissible routing layer M_j available for routing through that region, then the corresponding edge becomes available for M_j in G_j and hence in G_r^i. Otherwise, that edge will finally disappear making G_r^i with fewer edges before the next iteration of routing starts. Here, it is important to note that the next permissible layer M_j of M_i is the immediate layer above it for unreserved layer layer (ULM) and is not specific to the routing direction (orientation) such as vertical or horizontal. However, for reserved layer model (RLM), M_j will be two layers above of M_i i.e. M_j equals $M_2 + 2$ if it is within the permissible limit as per the prevailing fabrication process used for the design.

As the routing progresses, there are cases when many of the routing regions exhaust their permissible routing layers, making both G_j and G_r^i gradually sparse. This

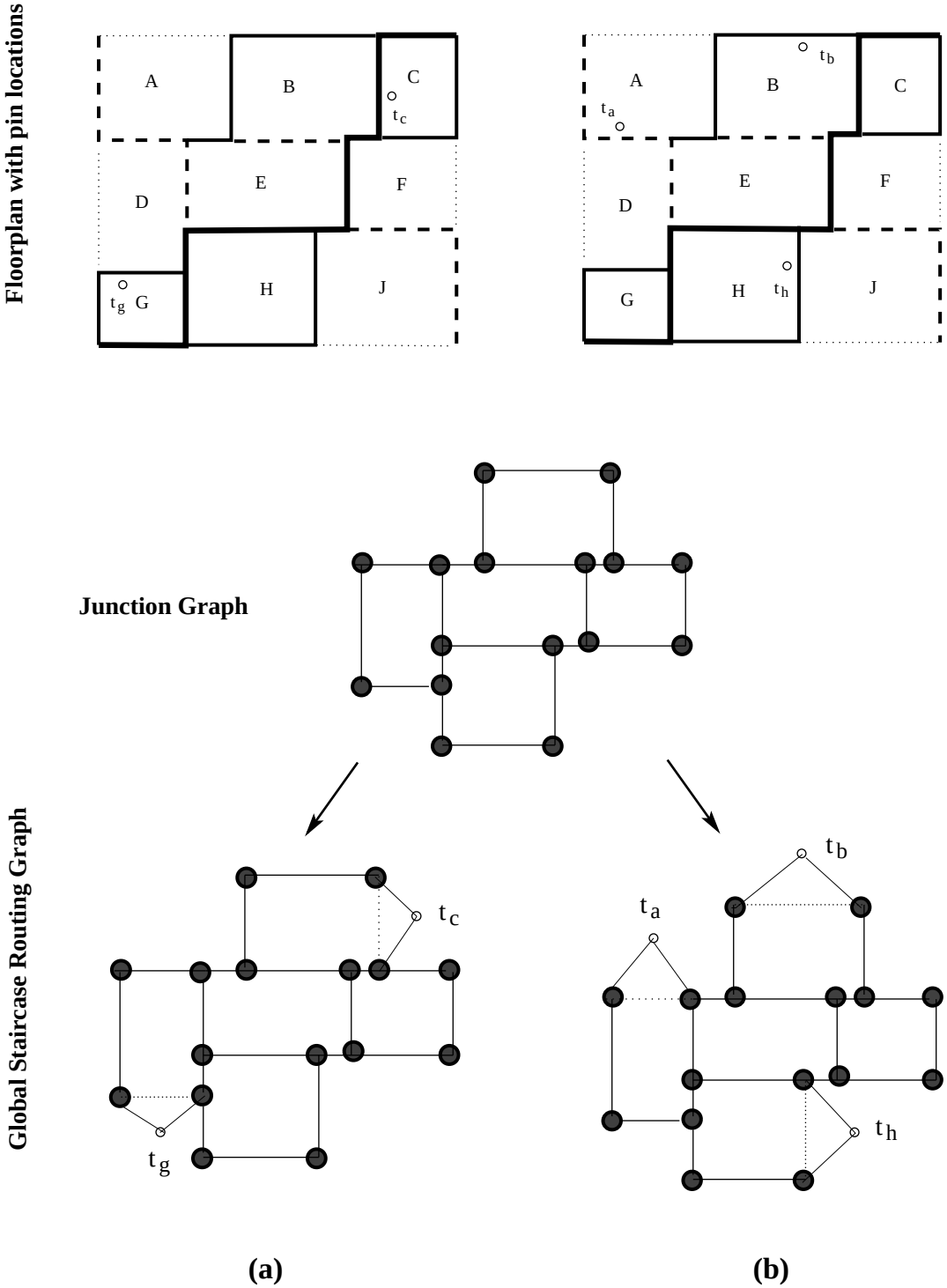


Figure 4.7: Construction of Global Staircase Routing Graph (GSRG) from Junction graph for a (a) 2-terminal net $n_i = \{t_c, t_g\}$, and (b) 3-terminal net $n_i = \{t_a, t_b, t_h\}$

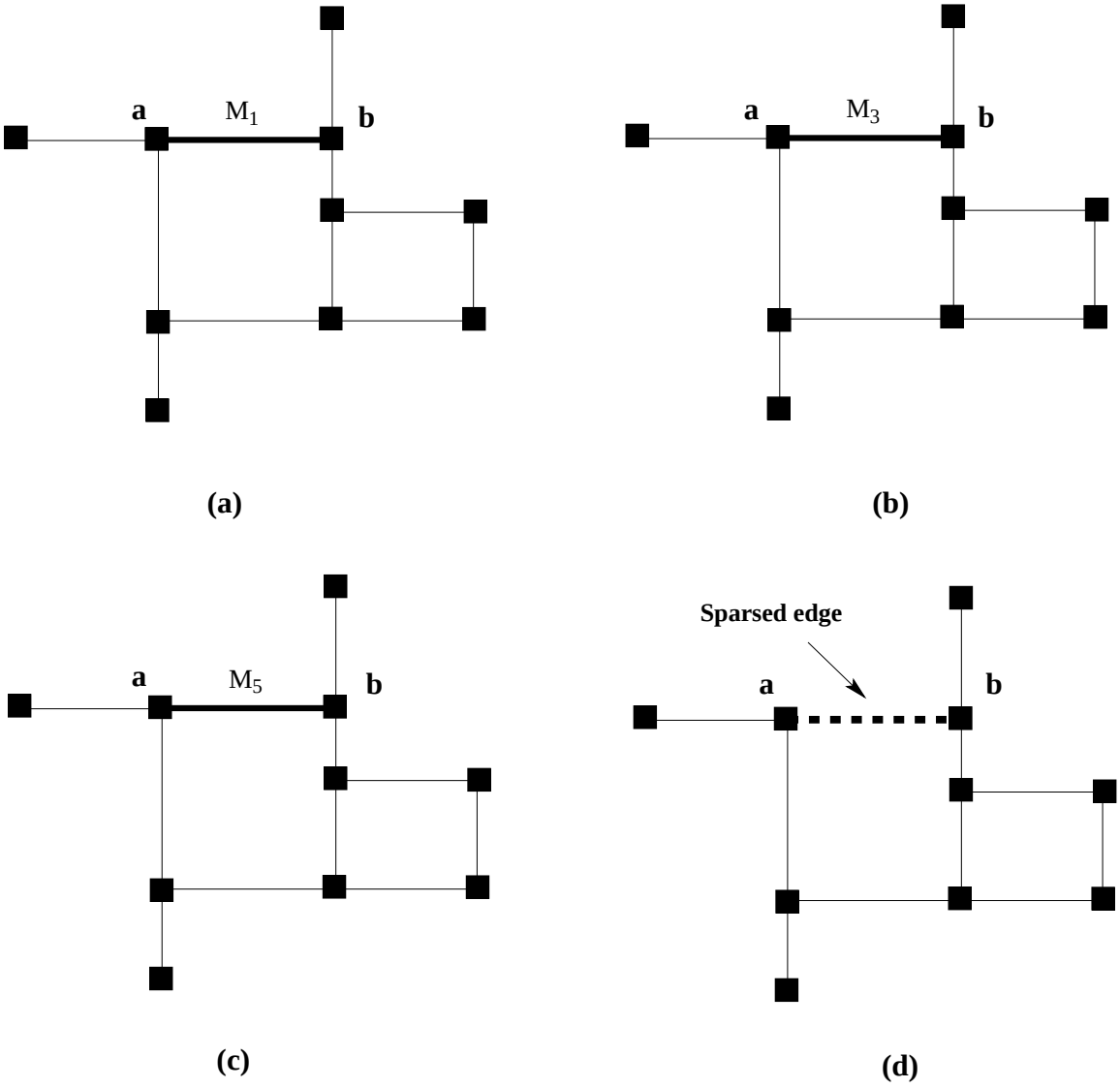


Figure 4.8: Sparsity of a horizontal (in bold) edge in a junction graph (assuming M_1, M_3 and M_5 as the permissible layers for routing)

situation is illustrated with the example presented in Figure 4.8 for a horizontal edge $\{a, b\}$ highlighted with a bold line between a pair of junctions a and b. In this example, we assume that edge $\{a, b\}$ allows M_1, M_3 and M_5 as the permissible routing layers. Once this edge consumes the entire routing capacity in each of these layers, it ceases to exist in the graph (with a cost equal to Infinity) as illustrated in Figure 4.8 (d). At this point, it is important to note that the routing capacity across the layers may remain uniform or may gradually decrease towards higher routing layers due to increasing wire width (see subsequent section for more details). In summary, the con-

gestion p_{s_k} in this work is constrained to a maximum value of 100% in each routing layer, thus restricting the number of routed nets (routing demand $uCap$) through a given segment to its capacity ($rCap$).

4.3.2 Multi-terminal Net Decomposition

In a global routing framework, routing a $t(>2)$-terminal net is crucial and obtaining an efficient solution for minimal length is a hard problem. Several works such as *Rectilinear Steiner Minimal Tree* (RSMT) topology [10] have been proposed in order to obtain the best possible $t-1$ net segments for a t-terminal net based on a well defined grid structure known as *Hanan grid* [9, 17]. Since the proposed global routing framework is based on a gridless graph model and the routing regions are aligned with the MIS/MDS, we can not adopt any such grid-based RSMT framework such as *FLUTE* [10].

Therefore, we propose a new method for multi-terminal net decomposition suitable for the proposed framework. We construct a complete undirected graph for a given t-terminal ($t > 2$) net $n_i \in N$, $G_c^i = (V_c^i, E_c^i)$ such that $V_c^i = \{t_k\}$, $\forall t_k \in n_i$ and $E_c^i = \{\{t_j, t_k\} \mid \forall t_j, t_k \in n_i \text{ and } t_j \neq t_k\}$. The weight of each edge $e_{jk} = \{t_j, t_k\} \in E_c^i$ is computed as *half the perimeter length* (HPWL) of the bounding box for each terminal pair (t_i, t_j) (see Figure 4.9 (a)). It is evident that $|V_c^i| = O(t)$ and $|E_c^i| = O(t^2)$. By employing $O(n^2)$ Prim's Minimum Spanning Tree (MST) algorithm [109], we obtain a *minimum spanning tree* (MST) T_c^i for G_c^i having $t-1$ edges, i.e., $t-1$ valid 2-terminal pairs. For each edge $e_{jk} = \{t_j, t_k\} \in T_c^i$, we perform 2-terminal net routing by applying Dijkstra's shortest path algorithm [109]. Once we obtain the routing paths for all such terminal pairs, we obtain the *Steiner points* by identifying the common routing segments as illustrated by an example in Figure 4.9.

In Figure 4.9, we consider an example of a 3-terminal net n_1 having terminals $\{t_a, t_b, t_c\}$ to illustrate the proposed net decomposition method. In this case, G_c^1 is a 3-clique with three vertices $\{t_a, t_b, t_c\}$, and three edges $\{t_a, t_b\}$, $\{t_b, t_c\}$ and $\{t_a, t_c\}$, along with their corresponding weights (see Figure 4.9 (a)). Only one of the instances of MSTs T_c^1 is greedily obtained by the proposed method as the final solution. Depending on the MST T_c^1 obtained, the proposed 2-terminal net segment routing, presented in the next section, on each valid terminal pair is applied. Once the routing for all the designated terminal pairs are obtained, we identify the *Steiner points* similar to the state-of-the-art grid-based multi-terminal net decomposition methods (FLUTE [10]),

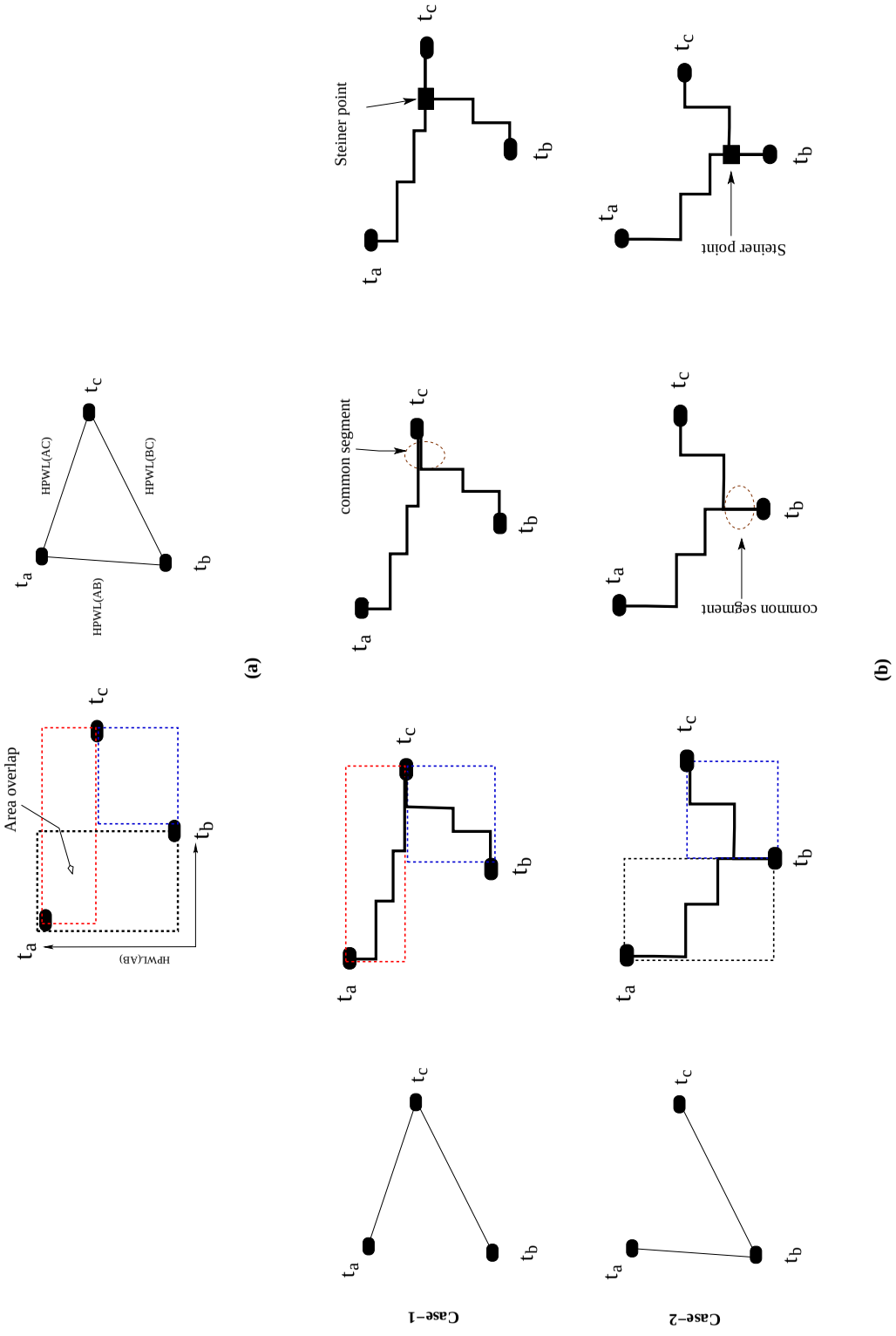


Figure 4.9: Illustrating the proposed Multi-terminal net decomposition

as illustrated in Figure 4.9 (b). This routing topology may be termed as *Staircase Minimal Steiner Topology* (SMST).

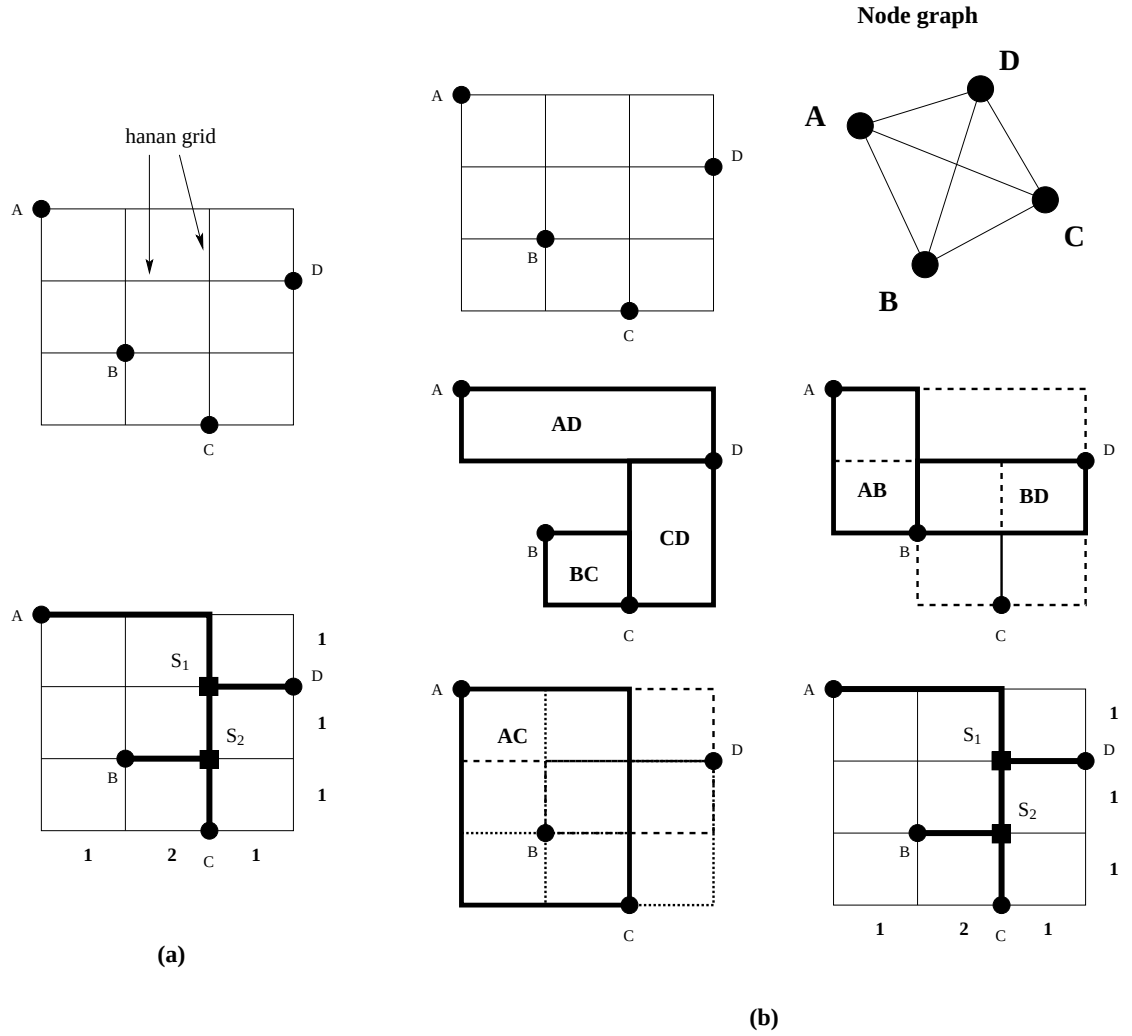


Figure 4.10: A comparative study betweem (a) Hanan Grid [9] based RSMT topology [10], and (b) our proposed Steiner topology (SMST)

In Figure 4.10, we present a comparative study between the traditional RSMT construction based on hanan grid [9] and the steiner tree topology we plan to propose. For the sake of simplicity, we assume the hanan grid to be uniformly spaced. Figure 4.10 (a) shows a net bounding box containing four pins $\{A, B, C, D\}$ and the corresponding 4×4 uniformly spaced hanan grid, assuming each horizontal/vertical grid separated by one unit length. A steiner tree solution for the corresponding pin topology is also shown in this diagram. In Figure 4.10 (b), we take the same pin topology and construct the 4-Clique. For each edge of the clique, we overlay the cor-

responding bounding box for each terminal pair one by one. When all these bounding boxes are overlaid, we obtain a similar hanan grid structure as in Figure 4.10 (a). We also obtain a identical steiner tree topology for all three valid terminal pairs $\{A, D\}$, $\{B, C\}$, and $\{C, D\}$. Their common segments yield the steiner points S_1 and S_2 (marked as ■). This example also validates that for a t pin net, a maximum $t - 2$ steiner points are possible to have on the steiner topology.

We also study the similarities between the blockage aware steiner tree topology obtained from a simple blockage free topology and our method in Figure 4.11. In this example, it is to be note that the hanan grid is not uniform. While (a) shows that one segment is running over a routing blockage (say a macro block), (b) depicts a modified blockage aware topology but with an additional wirelength penalty. The latter corresponds to the topology we proposed in this work (depicted by both the instances in Figure 4.9).

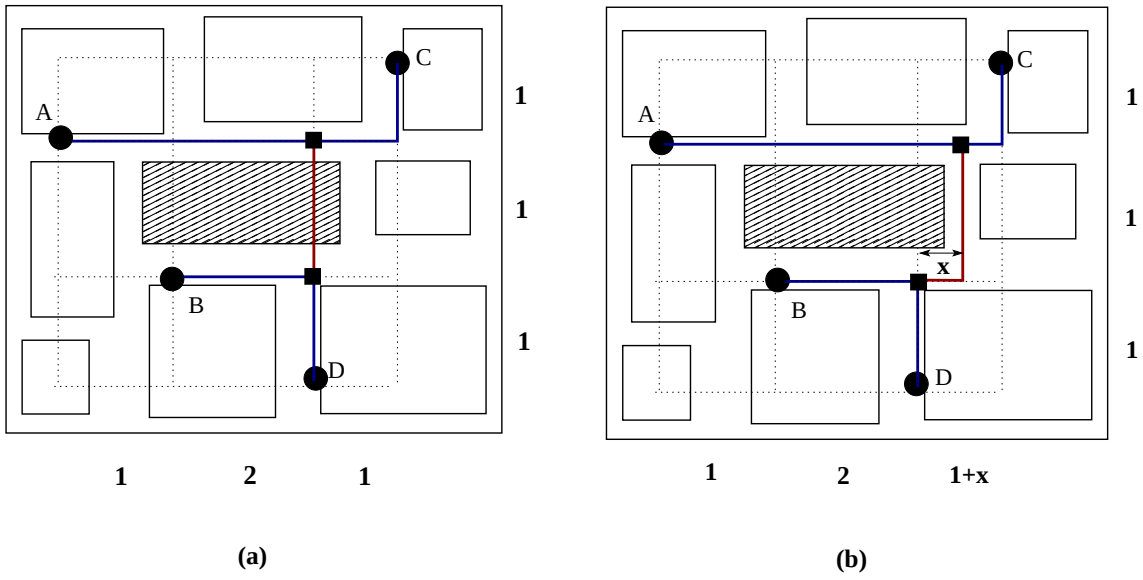


Figure 4.11: A layout with a net containing pins $\{A, B, C, D\}$ depicts: (a) a simple RSMT topology [10], and (b) our proposed steiner toploogy (similar to blockage aware topology)

4.3.3 Net Ordering

In sequential global routing frameworks, the order in which the nets are routed plays a crucial role towards successful routing completion, wirelength, via count and the congestion. Usually, shorter nets are routed first before the longer nets. Therefore,

a suitable net ordering has to be done before the routing starts. In the proposed routed, we obtain an routing order of the nets by utilizing the information available in the corresponding floorplan bipartition hierarchy, MSC tree. At any given level of the MSC tree, we order the nets based on the non-decreasing order of HPWL followed by NetDegree. However, for the entire hierarchy, we route the nets in a level wise fashion. We consider the nets in the lower most hierarchy the first, then subsequently going up. As evident from the MSC tree, the nets cut at the same level nodes are more likely to be disjoint if they are not already cut at the higher nodes. This has the potential to reduce the number of ripup and reroute during several iterations of optimization in global/detailed routing.

4.3.4 The Algorithm

We present the proposed early global routing algorithm, namely *STAIRoute*, in Algorithm 4. This algorithm takes two inputs, namely an ordered set of nets N and the junction graph G_j for a given floorplan F. For each net $n_i \in N$, the corresponding routing graph G_r^i is constructed and a routing path for n_i is identified by applying a shortest path algorithm on G_r^i. In this method, an $O(n^2)$ implementation of Dijkstra's single source shortest path algorithm [109], namely *DijkstraSSP()* is used for identifying a routing path between a pair of net pins (terminals). For 2-terminal nets, we directly apply *DijkstraSSP()* for identifying the corresponding routing path. In order to identify a shortest route for a multi-terminal net ($t > 2$), we first apply the multi-terminal net decomposition method presented in Section 4.3.2 on each net for identifying $t-1$ valid pairs of terminals. We then apply *DijkstraSSP()* for each valid terminal pair and subsequently identify the steiner topology of the net.

Theorem 2 *Given a floorplan having n blocks and k nets having at most t-terminals ($t \geq 2$), the algorithm STAIRoute takes $O(n^2kt)$ time.*

Proof As per Lemma 8, GSRG construction for a t pin net takes $O(t)$ time. We consider two instances of net routing:

(a) for each 2-terminal net routing, our implementation of Dijkstra's single source shortest path algorithm (DijkstraSSP) takes $O(n^2)$; and

Algorithm 4 STAIRoute

Inputs: $G_j(V_j,E_j)$, Ordered nets N
Outputs: Global routing for each t-terminal ($t \geq 2$) net ($n_i \in N$) with 100% routability and usage $\leq$ 100%

for all sorted nets $n_i \in N$ **do**
  G_r^i = ConstructGSRG(G_j,n_i)
  if Netdegree(n_i) == 2 **then**
    /*Netdegree(n_i) = Number of terminals in n_i*/
    Source = IdentifySource(n_i.terminals) /*for Forward or Backward search (see Figure 4.15)*/
    Path(Source,Sink) = DijkstraSSP(G_r^i,Source)
    if There exists a routing path from Source to Sink **then**
      n_i is routed.
      Update *uCap* for the respective routing segments.
      NetLength(n_i)
      ViaCount(n_i)
    else
      Routing n_i is a failure and continue for n_{i+1}
    end if
  else
    G_c^i = ConstructNodeClique(n_i.terminals)
    T_c^i = ObtainMST(G_c^i)
    for all edges $(t_j, t_k) \in T_c^i$ **do**
      Source = IdentifySource(t_j, t_k) /*for Forward or Backward search (see Figure 4.15)*/
      Path(Source,Sink) = DijkstraSSP(G_r^i,Source)
      if There exists a routing path from Source to Sink **then**
        2-terminal net segment is routed; calculate the segment length.
        update the *uCap* for the respective routing segments.
      else
        Routing n_i is a failure and continue for n_{i+1}
      end if
    end for
    Identify the *Steiner Point(s)* /*see Figure 4.9*/
    NetLength(n_i)
    ViaCount(n_i)
  end if
end for
Return the values of routing completion, and total wirelength, congestion and via count for all the nets.

(b) for each t-terminal ($t > 2$) nets, we first compute a complete graph G_c^i for all the terminals in the net, and running minimum spanning tree (MST) algorithm on G_c^i takes $O(t^2)$ for identifying the steiner topology. In our version of Prim's minimum spanning tree (MST) algorithm also takes $O(t^2)$. Subsequently for each terminal pair (t_i, t_j) representing an edge in the MST T_c^i of G_c^i, we obtain the single source shortest routing path between these two terminals using DijkstraSSP in $O(n^2)$ time.

Thus, for each $t(\geq 2)$ terminal net, STAIRoute has the worst case time complexity as $O(t + t^2 + n^2 t)$, i.e., $O(n^2 t)$, since a given net may be connected to all n blocks resulting in $t = n$ in the worst case. Typically, the number of terminals t in a net is very small as compared to n, i.e., $t = o(n)$. Therefore, the worst case time complexity of STAIRoute for routing all k nets is $O(n^2 kt)$. □

With a better implementation of Dijkstra's single source shortest path algorithm in $O(n \log n)$ time [109], the worst case time required by STAIRoute for routing k nets is $O(nkt \log n)$. Further improvement can be achieved in the runtime of STAIRoute by implementing the Prim's algorithm in $O(t \log t)$, although the worst case time complexity for STAIRoute remains $O(nkt \log n)$.

In practical circuits, the number of pins (terminals) is 2 for majority of the nets, thereby keeping the average net degree of all the nets usually within a very small constant number say 4. Therefore, the effort on multi-terminal net decomposition by the proposed approach (see Figure 4.9) is minimal, considering $t = O(1)$, leading to the worst case time needed for STAIRoute to $O(nk \log n)$. However, for any general purpose circuit with no such consideration, $O(nkt \log n)$ is the worst case time required for all the k nets to be routed in a floorplan containing n blocks. We illustrate the working of the proposed early global routing of a 2-terminal net and a 3-terminal net in Figure 4.12 (a) and (b) respectively.

4.4 Further Improvements in STAIRoute

In the previous section, we discussed the global routes of a set of nets through a set of monotone routing regions across multiple metal layers, employing both reserved and unreserved routing model. In unreserved layer model, the net segments pass through a horizontal/vertical routing region in any routing layer without having any preferred routing direction; this means an wire segment can be routed in a particular metal

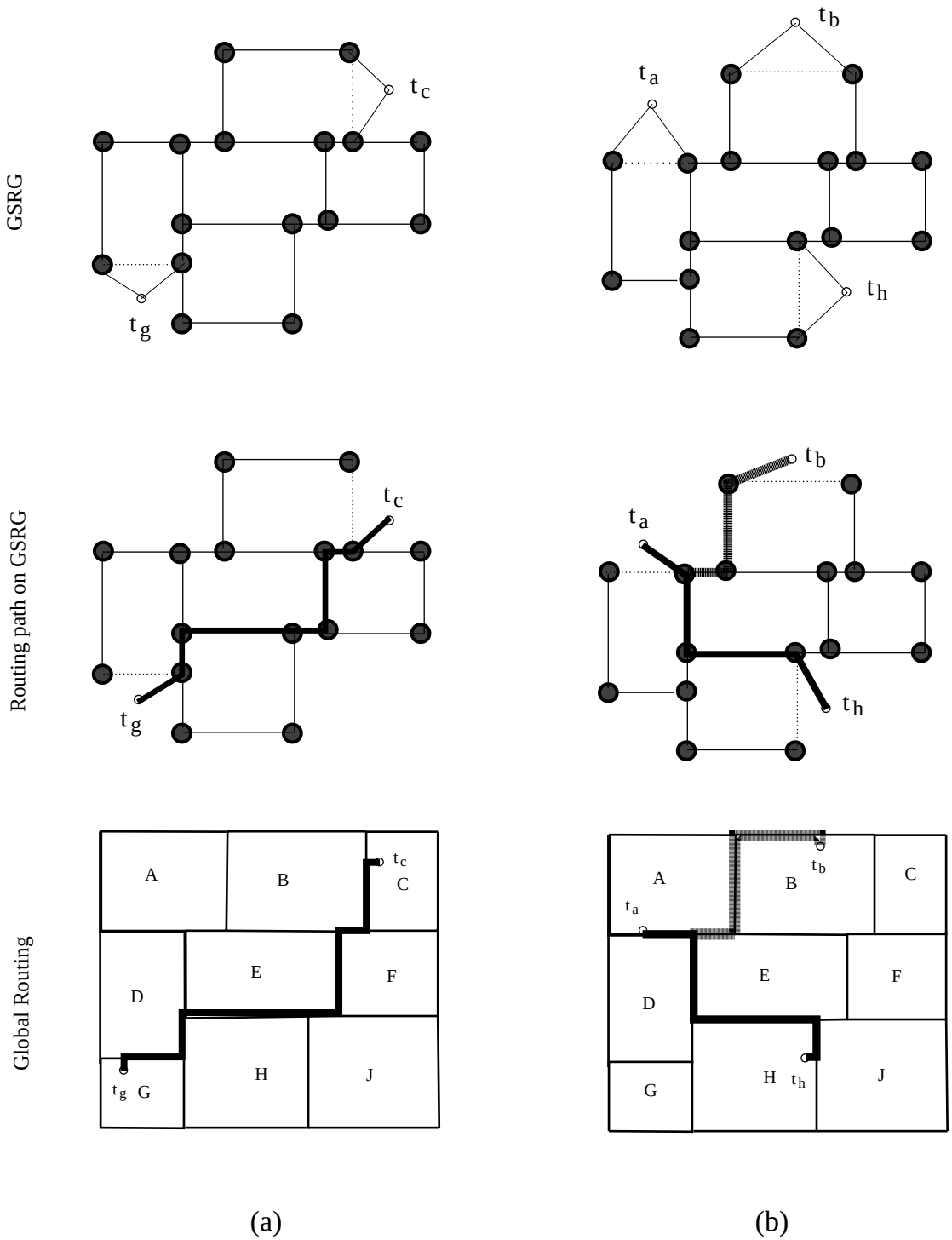


Figure 4.12: Illustration of early global routing of a (a) 2-terminal net $n_i = \{t_c, t_g\}$, and (b) 3-terminal net $n_i = \{t_a, t_b, t_h\}$ in a given floorplan

layer using either horizontal or vertical routing. On the other hand, reserved layer model strictly ensures the routing of a vertical (horizontal) segment of a net through the designated layers; for example a horizontal routing segment is routed through odd metal layers such as $M_1, M_3, \cdots$, while the vertical segments are allowed in even layers only i.e. $M_2, M_4, \cdots$.

So far in the proposed router, we assumed the capacity of the routing regions to be uniform across the routing layers. This model do not support any wire width variation across the routing layers. But the technological advancement in IC fabrication technology has enabled the designers to route the nets using different metal widths at different metal layers, as depicted in Figure 4.13. Therefore, the number of routing tracks through the same routing area is reduced;obviously, a particular routing region can route more number of narrower wires than wider ones.

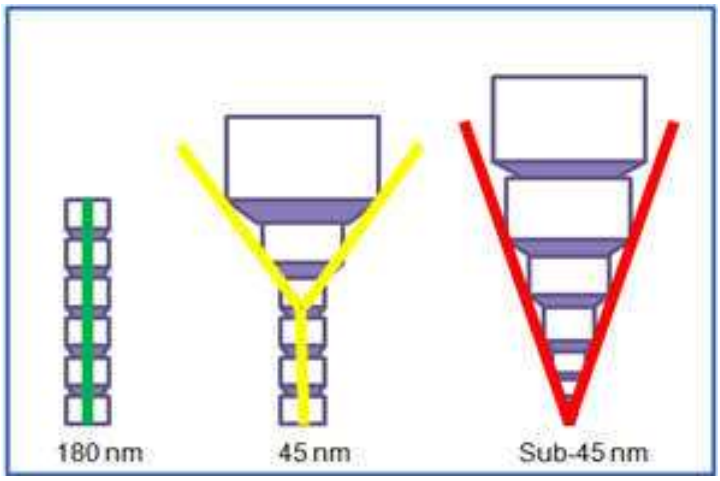


Figure 4.13: Metal width variation in different fabrication process nodes [11]

In Figure 4.13, we illustrate different scenarios of metal width variation across the metal layers in different IC fabrication technology nodes such as $180nm$, $45nm$, and those below $45nm$. From these instances, we can clearly identify specific variation patterns. For example, in $180nm$ node, it depicts a uniform wire width across the routing layers implying the routing capacity to remain the same in higher layers as in the lower layers. However, $45nm$ node and the nodes below it for instance, show different width variation patterns similar to a piece-wise linear (PWL) pattern and a more aggressive linear pattern respectively. Hence, the capacity of the routing regions in designs using $45nm$ node shall have uniform capacity from the lowest possible layer up to a certain layer, followed by scaling down the capacity similar to the steps of a

ladder due to a step wise variation in metal width. Finally, the design using nodes below $45nm$ are entitled to have hyperbolic capacity variation across the layers due to almost linear increase in width with the metal layers. This gives us the motivation to study the scaling effect of the routing capacity of the regions across the layers on the proposed global routing method. This study shows how this routing model can adapt to practical routing challenges like one of the aggressive metal width variation, as this is being mostly used in the existing state of the art academic and industrial global routers.

4.4.1 Layer wise Routing Capacity Scaling

In this routing model, we realized layer assignment of the nets for multiple metal layers using a parameter called *currLayer(s_k)* associated with each segment s_k. This initialized to M_1 in case of unreserved layer model and M_1/M_2 for horizontal/vertical orientation in reserved layer model. For both the models, the maximum permissible metal layer is a technology aligned parameter M_{max}. When congestion is about to occur in s_k ($p_{s_k} = 1$), we increment *currLayer(s_k)* to the subsequent metal layer. Here the subsequent metal layer has different implication in (un)reserved layer model; the subsequent layer can either be one layer above the *currLayer(s_k)* or the next permitted layer based on the particular (horizontal/vertical) orientation of s_k in the respective unreserved/reserved layer models. This means that the region s_k has exhausted all of its routing capacity (i.e. $uCap_{s_k} = rCap_{s_k}$) for the current metal layer and is now ready for routing the nets through it for the next metal layer restricted by M_{max}. In this regard, the variation of $rCap_{s_k}$ across the metal layers (up to M) plays a significant role and thus directly impacts the routing completion of all the nets.

In Figure 4.14 (a), we present different scenarios of routing capacity variation for a given routing region s_k across multiple metal layers. In case of *uniform* profile, the routing capacity $rCap_{s_k}$ of region remains the same across the metal layers. The other two capacity profiles are: a *hyperbolic* (or $1/M$) pattern and a *ladder* pattern. In case of the hyperbolic pattern, $rCap_{s_k}$ is more aggressively scaled across the metal layers, the latter has a relatively relaxed trend of metal pitch/width (hence routing capacity) variation across the metal layers (see Figure 4.13 [11]). Both the capacity variation profiles have great impact not only on the routing completion, but also on the via count as more routing layers are used to route the nets. Notably, this effect has less significant impact on netlength, as our router always try to confine the routing within

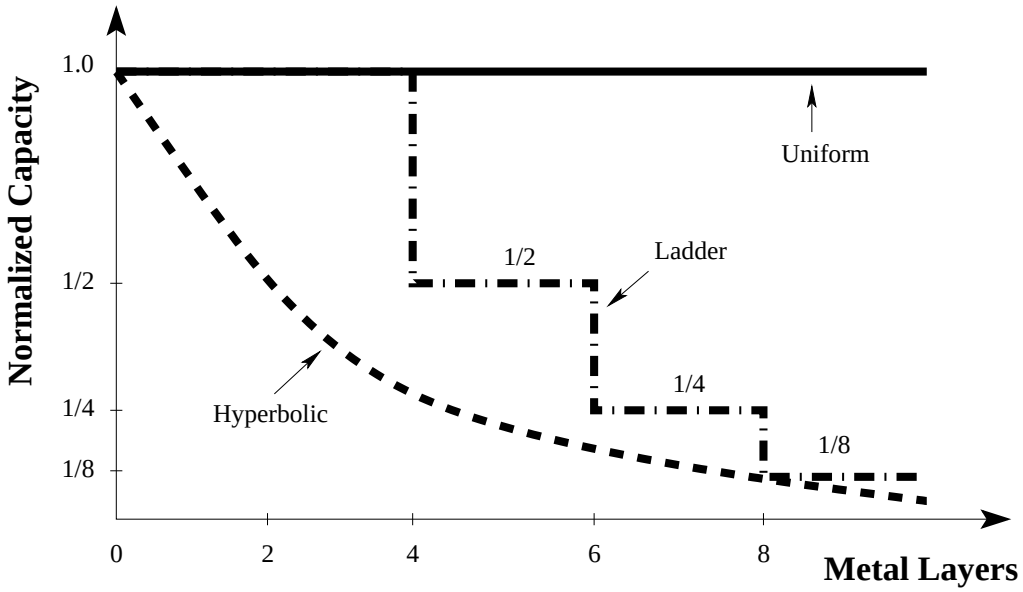


Figure 4.14: Routing capacity (normalized with respect to M_1 routing layer) profiles of a routing region vs metal layers

the bounding box of the net terminals (pins) using monotone staircase patterns.

In our routing model, the congestion in a routing region is measured as the ratio of the routing demand and the routing capacity and has no direct impact on it. However, it has an indirect effect on it due to the overall routing completion, as the proposed router tries hard to complete the routing of as many nets as possible in lower layers than in higher layers. In summary, varying degree of impact due to capacity scaling in the upper layers can be seen on the overall global routing metrics depending the number of routing layers used. This has the potential to dominate the quality of the early global routing solution across different designs, different floorplan instances of a design and different bipartitioning results of the same floorplan instance due to different values of the trade-off parameters (refer to Section 5).

4.4.2 Directional Routing Path

For each 2-terminal net (segment), we consider two cases of identifying the source vertex between a pair of terminals before we apply the shortest path algorithm as:

1. the minimum x coordinate (or the minimum y coordinate in case both the terminals have the same x coordinate)

2. the maximum x coordinate (or the maximum y coordinate in case both have

the same x coordinate)

and the procedure *IdentifySource()* in Algorithm 4 is used for that purpose. In the first version of the proposed router, we used only the first option (a source with lower x coordinate). Subsequently, we consider both the options to identify the route of a net that are potentially different in similar congestion scenario and the adjacency list of the vertices in GSRG. When the congestion scenario in the entire layout is highly skewed, leading to a route that is highly possible to be different. Our study has shown that, this directional search (see the results in Section 4.5) can be advantageous in many cases such as different designs, different floorplan instances of a same design and so on.

We call these modes as *Forward* (FWD) and *Backward* (BACK) search respectively. In Figure 4.15, we illustrate these cases for a 2-terminal net $\{t_g, t_c\}$ and show that both search procedures can potentially give different routing paths depending on the local congestion scenario. One search mode may have a potentially better solution than the other in terms of routability, congestion scenario along with net length, and also the via count. The variation in net length due to FWD (BACK) search arises when certain region(s) along the respective paths are fully utilized in a given metal layer; with the possibility of switching to the next available metal layer if permitted, leads to increase in the via count. Otherwise, the routing path is detoured beyond the bound box of the terminals, leading to an increase in wirelength. As long as the alternatives paths remain confined within the bounding box of the terminals, no variation in netlength can be expected among these cases.

In the unreserved layer model, routing of a net incurs a number of vias due to difference in the metal layers used to route through the corresponding routing resources. It does not depend on their vertical/horizontal orientation. In case of reserved layer model, the number of vias along a routing path depends on the number of bends in it, i.e., the alternating (horizontal/vertical) orientation of the contiguous routing resources, for a minimum change of one metal layer among the resources along that path [22]. Congestion in channels may also contribute to the number of vias along a routing path, in both the cases. From the example shown in Figure 4.16 (a) and (b), we notice that the routing path for a given net (t_g,t_c) needs 3 and 5 vias for FWD and BACK searches respectively. Therefore, depending on the netlist and the floorplan topology of a given circuit, one method may dominate over the other. This method can be extended to t (> 2)-terminal nets, since we decompose those nets using the

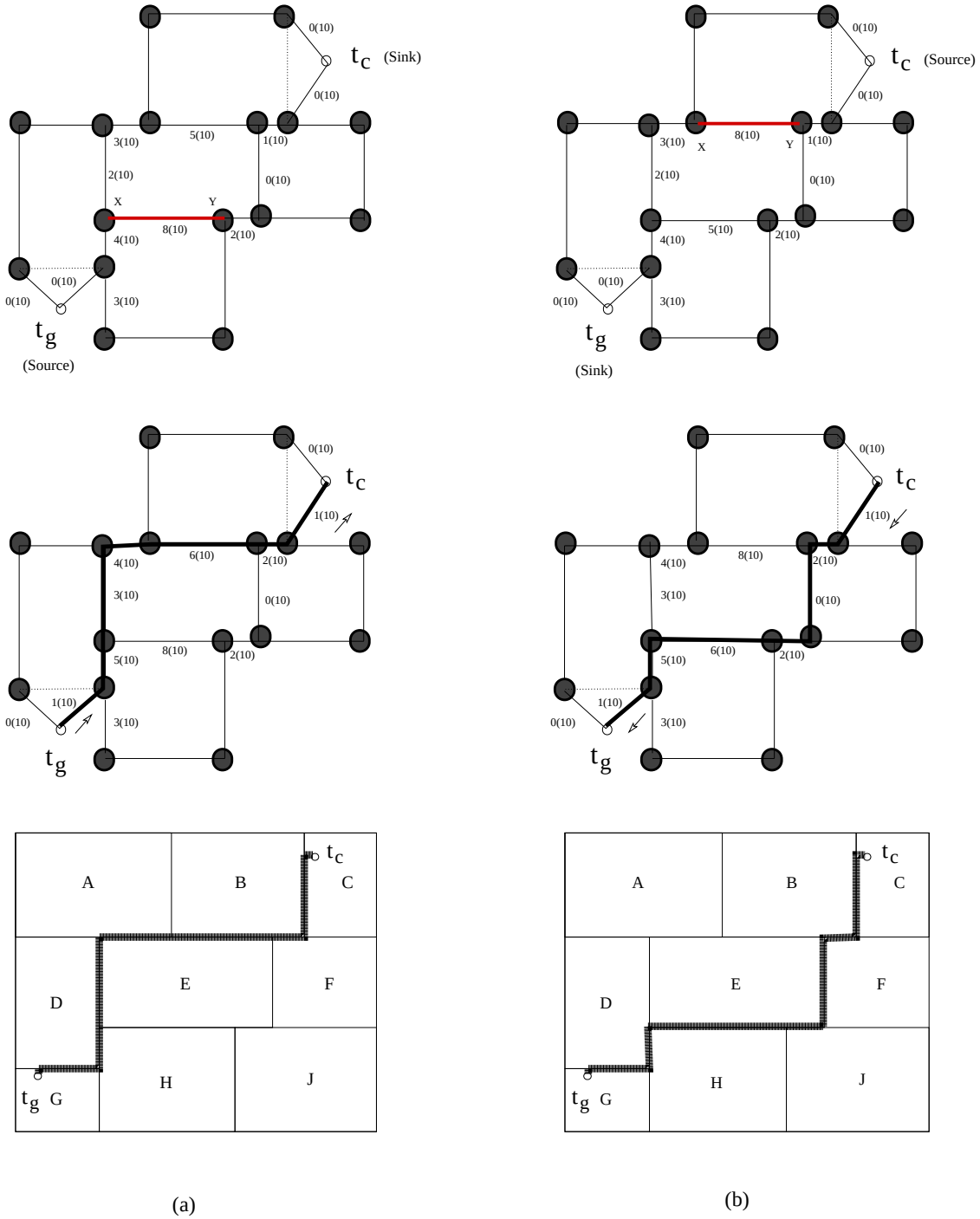


Figure 4.15: A routing path for 2-terminal net (t_g,t_c) obtained by: (a) forward Search, and (b) backward Search

method stated earlier into 2-terminal net segments and a better routing path for each of the resulting net segments can be obtained while employing either of the search

procedures at a time.

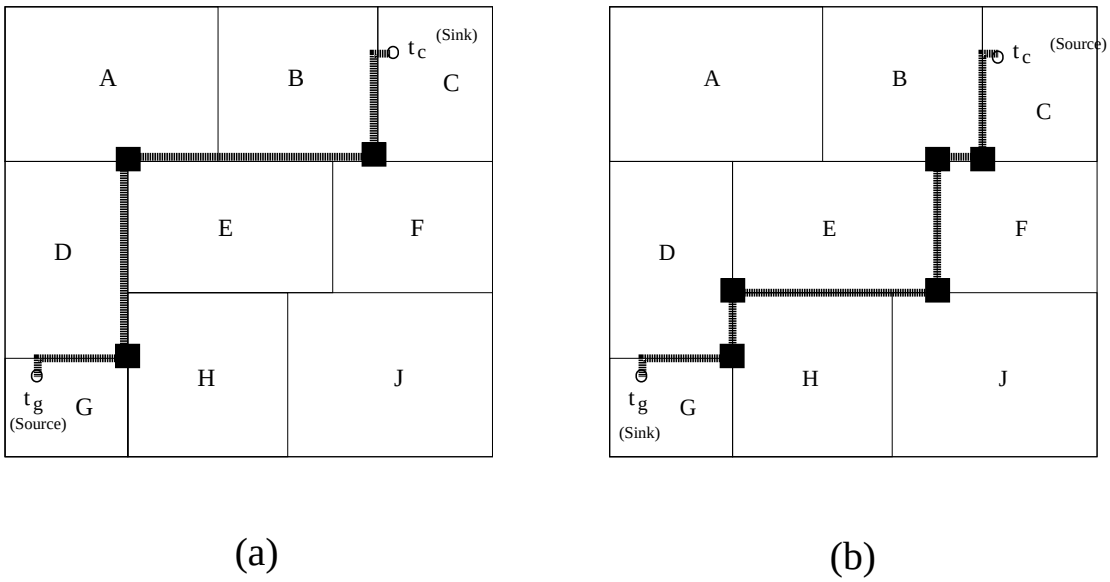


Figure 4.16: Impact on via count for: (a) forward Search with 3 vias, and (b) backward Search with 5 vias

4.5 Experimental Results

In this section, we validate the proposed early global routing method *STAIRoute* on MCNC and GSRC floorplanning benchmark circuits summarized in Table 4.1, also presented in Chapter 3.5 [14, 113]. The algorithm and its improvements presented in this chapter were implemented in C programming language and the experiments were run on a Linux platform (2.8GHz, 4GB RAM). In this experimental setup, the floorplan instances for each of the benchmark circuits were generated with a random seed using *Parquet* Floorplanning tool [14, 113].

4.5.1 Unreserved Layer Routing for BFS method

Since STAIRoute is the first early global routing method after floorplanning, and does not fall in the purview of the existing post-placement global routing methods, we can not compare the routing results with them directly. Moreover, the recent global routing benchmarks used by those post-placement global routers do not have any corresponding floorplanning benchmarks. Instead, we compare the routed length

Table 4.1: MCNC and GSRC Floorplanning Benchmark Circuits [14]

Suite	Circuit	#Blocks	#Nets	Avg. NetDeg
MCNC	apte	9	44	3.500
	hp	11	44	3.545
	xerox	10	183	2.508
	ami33	33	84	4.154
	ami49	49	377	2.337
GSRC	n10	10	54	2.129
	n30	30	147	2.102
	n50	50	320	2.112
	n100	100	576	2.135
	n200	200	1274	2.138
	n300	300	1632	2.161

of each t-terminal ($t > 2$) net obtained STAIRoute with the corresponding Steiner length computed by the state-of-the-art academic Steiner topology generation tool *FLUTE* [10]. In this experiment, we obtain the Steiner topology of a net using FLUTE without considering any routing blockage or any kind of initial congestion map. The results obtained in our first experimental setup by STAIRoute used the floorplan bipartitioning results obtained by GenMSCut (see Chapter 3) for $\gamma = 0.4$ and $\epsilon = 0.05$ (see the corresponding bipartitioning results in Table 3.2 obtained for area balanced mode) presented in Table 4.2. In this experiment, STAIRoute used unreserved layer model for up to 2 metal layers to route the nets. The congestion model (see Equation 4.1 and 4.2) has ensured that no congestion takes place in any of the monotone staircase routing regions yielding 100% routing completion for each circuit.

In this table, we summarize the results obtained for *Runtime*, *Routability (%)* and *net length* for each benchmark circuit in Table 4.1. These results show that *routed netlength* obtained by STAIRoute is comparable to *Steiner length* computed by FLUTE [10] as the average *NetDeg* for each GSRC benchmark circuit is slightly higher than 2. On the other hand, due to higher average NetDeg (maximum 4.15 for *ami*33) in MCNC benchmark circuits, the routed netlength are slightly higher than the corresponding Steiner length. It a shows that Netlength to Steiner length ratio (L/F) has a maximum value around 1.15 for some of the circuits, while the lowest value can go as low as 1.05 for *apte* and 1.06 for $n300$. Our study on the existing post-placement global router showed that Netlength to Steiner length ratio

Table 4.2: Summary of the routing results using Unreserved Layer Model (ULM) for up to 2 metal layers

Circuit	Runtime (sec)	Routability (%)	Netlength (μm)		Length Ratio (L/F)	#Via
			Routed (L)	Steiner [10] (F)		
apte	0.192	100	397447	376652	1.055	0
hp	0.200	100	284601	245801	1.158	2
xerox	0.316	100	688107	633533	1.086	0
ami33	0.744	100	161636	142748	1.132	4
ami49	2.012	100	1794979	1629255	1.101	0
n10	0.164	100	18505	16626	1.113	0
n30	0.444	100	56475	49370	1.143	0
n50	1.372	100	144451	125018	1.155	4
n100	6.748	100	237653	214578	1.107	6
n200	43.991	100	410849	381021	1.078	12
n300	104.382	100	744416	699006	1.064	2
Average Length Ratio					1.108	-

has also been found to be as low as 1.04 (results in the subsequent chapters) and hence comparable to some of the results obtained here. The last column contains the estimated number of vias used for each of the circuits using ULM. It is important to note that we obtained all the above results by restricting ourselves up to 2 metal layers. We see that while some of the circuits return zero via count. This implies that the routing of all the nets are confined in the first metal layer M_1 only, while the other instances with non zero via count indicates routing takes place through two metal layers (M_1, M_2) due to 100% usage of the routing capacity in M_1 in certain routing regions. The runtime values for routing show that it is increasing with the number of blocks n and nets k as per Theorem 2 and does not include the runtime for the corresponding bipartitioning methods.

4.5.2 Reserved Layer Routing for BFS/DFS methods

Now, we compare the routing results obtained by STAIRoute using the corresponding bipartitioning results obtained by GenMSCut and GenMSCut_DFS presented in Chapter 3 respectively. These experiments were conducted on MCNC and GSRC benchmark circuits for $\gamma = 0.4$ only. We tag the corresponding routing results as BFS-NB and DFS-NB respectively, as they do not consider bend minimization in their bipartitioning objective function. For the corresponding bipartitioning results for BFS-NB and DFS-NB, please refer to Figure 3.10) in Chapter 3.

In Table 4.3, we summarize the early global routing results obtained by STAIRoute for (a) Netnegth (μm), (b) via count and (c) the worst average congestion [15]. It is to be noted that all these results correspond to 100% routing completion while using maximum up to 8 metal layers using reserved layer model (RLM). From these results, it is noticeable from the normalized geometric mean values that BFS-NB mode gives slightly better routed netlength and via count. The average of worst congestion obtained for BFS-NB is also marginally better than that in case of BFS-NB method, even for the most of the circuits. The first observation is that netlength for both the modes remain within 20% additional Steiner length. Another important observation here is that the worst congestion in any of the routing regions for any circuit does not go beyond 100%, with a maximum value of the worst congestion going around 96% and 97% for BFS-NB and DFS-NB respectively. All these results correspond to 100% routing completion for all γ values and the floorplan instances of all the benchmark circuits.

Table 4.3: Comparing the routing results for reserved layer model (RLM) using $\gamma = 0.4$ and 8 metal layers: between BFS-NB and DFS-NB

Mode	NetLength		**Steiner**	#Via		Worst Congestion	
Circuit	BFS-NB	DFS-NB	Length [10]	BFS-NB	DFS-NB	BFS-NB	DFS-NB
apte	429957	427688	374756	429	429	0.873	0.881
hp	229891	234680	162799	453	455	0.958	0.973
xerox	1122511	1141519	1008712	1235	1273	0.641	0.675
ami33	130042	130252	110983	1181	1209	0.950	0.932
ami49	1904592	1887049	1678283	3535	3547	0.819	0.801
n10	23494	23393	18732	235	227	0.766	0.689
n30	64718	64353	53236	992	1006	0.733	0.770
n50	173593	173924	144719	3110	3295	0.875	0.932
n100	262572	263828	223822	7218	7055	0.909	0.928
n200	665756	661387	588080	19290	19405	0.800	0.812
n300	882914	882790	773875	31543	31596	0.889	0.866
(Norm.) Geo. Mean	**1.188**	1.189	1.000	**0.992**	1.000	**0.832**	0.836

We extend this study on the early global routing metrics such as routed netlength, via count, congestion and routability for the given benchmark circuits. We obtain these results by running STAIRoute for 8 metal layers using reserved layer model using the bipartitioning results for both BFS-NB and DFS-NB modes with different values of $\gamma \in [0.1, 0.7]$. The corresponding routing results for both BFS-NB and DFS-NB methods of bipartitioning are presented in Tables 4.4, 4.5 and 4.6 for routed netlength, via count and worst congestion respectively. We noticed that 100% routing

completion is achieved for across all the modes, specified γ values and all the circuits.

In case of netlength vs γ results presented in Table 4.4, we obtain the routed netlength and Steiner length for all the nets for a given flooplan instance of a circuit versus γ. The Steiner length is computed by FLUTE [10] without considering any routing blockages or initial congestion and is used to normalized the corresponding netlength for both the modes. The normalized geometric mean values for all the circuits for each γ show that the netlength variation is negligible. Careful study on the netlength variation for individual circuits for different γ value show that the variation is noticeable towards higher γ values such as 0.6 and 0.7. This is due to the fact that for higher γ values emphasizes more on maximizing the area balance objective (with the corresponding weight of γ) than minimizing the net cut (due to the weights 0.4 and 0.3), in obtaining a bipartitioning solution (monotone staircase in this case) for both BFS and DFS based methods. We also notice that BFS-NB method is consistently more efficient in obtaining smaller netlength than DFS-NB for all γ values, specially for most of the circuits. Despite that, the normalized geometric mean shows that the routes of the nets are approximately 20% longer than the Steiner length (presented in the last column) in both the modes.

A similar study on the variation of via count for the specified γ values is presented in Table 4.5. For each circuit, we compute the average of all the vias obtained for different γ values and modes. This average via count (see the last column) is used to normalize the via count values for different γ values and modes in the respective column. The normalized geometric mean for each mode and γ show that BFS-NB mode has slightly better values over DFS-NB mode. Further study on the via count for each γ reveals that it follows a non-decreasing pattern with the increasing γ values, irrespective of the modes, for most of the circuits except $n100$ and $n200$. The similar observations can be made for netlength as well in Table 4.4.

The worst congestion values for each circuit in Table 4.6 also show that the bipartitioning results in BFS-NB mode yield better values of the worst congestion than those for DFS-NB mode. This is also evident from the geometric mean values in the last row.

4.5.3 Directional Routing and Capacity Scaling

We further extend the experiments on STAIRoute using reserved layer model using up to 8 metals layers, using the bipartitioning results obtained by BFS-NB method

Table 4.4: Comparing Netlength (μm) w.r.t γ: between BFS-NB and DFS-NB (in the bracket below)

γ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	Steiner
Mode/ Circuit	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	Length [10]
apte	429957 (427688)	429957 (427688)	429957 (427688)	429957 (427688)	429957 (427688)	429957 (426542)	429834 (419855)	374756
hp	229891 (234680)	229891 (234680)	229891 (234680)	229891 (234680)	229891 (234680)	229891 (234680)	229891 (234680)	162799
xerox	1122511 (1141519)	1122511 (1141519)	1122511 (1141519)	1122511 (1141519)	1122511 (1141519)	1125723 (1141519)	1125723 (1161381)	1008712
ami33	130042 (130252)	130042 (130252)	130042 (130252)	130042 (130252)	130042 (130252)	130268 (129223)	128664 (129471)	110983
ami49	1904592 (1887049)	1904592 (1887049)	1904592 (1887049)	1904592 (1887049)	1904592 (1887049)	1896780 (1886331)	1893587 (1886331)	1678283
n10	23494 (23393)	23494 (23393)	23494 (23393)	23494 (23393)	23494 (23393)	23494 (23393)	23494 (23393)	18732
n30	64718 (64353)	64718 (64353)	64718 (64353)	64718 (64353)	64718 (64353)	64815 (64353)	64740 (64303)	53236
n50	173593 (173924)	173593 (173924)	173593 (173924)	173593 (173924)	173593 (173924)	173371 (173909)	173307 (173683)	144719
n100	262572 (263828)	262572 (263828)	262572 (263828)	262572 (263828)	262572 (263828)	263676 (263644)	263625 (263644)	223822
n200	665756 (661387)	665756 (661387)	665756 (661387)	665756 (661387)	665756 (661387)	665310 (661563)	664465 (662550)	588080
n300	882914 (882790)	882914 (882790)	882914 (882790)	882914 (882790)	882914 (882790)	883750 (882044)	883560 (881287)	773875
Norm. Geo. Mean	**1.188** 1.189	**1.188** 1.189	**1.188** 1.189	**1.188** 1.189	**1.188** 1.189	**1.188** 1.188	**1.187** 1.188	1.000

on different floorplan instances of the same circuit. For a given benchmark circuit (see Table 4.1 for details), the *best case* (BC) and the *worst case* (WC) floorplan instances are designated in the terms of the total *half perimeter wire length* (HPWL) of all the nets as the ones with the smallest and the largest HPWL respectively among the floorplan instances generated for that circuit. In these experiments, we refer to Figure 4.14 for 3 different capacity scaling profiles due to varying metal pitch across the routing layers (see Figure 4.13). We also refer to Figure 4.15 for two possible directions of routing in order to explore different routing paths with potential congestion minimization and fewer via counts. In forward search, the source and the sink definition between a pair of terminals (net pins) for identifying Dijkstra's shortest path on GSRG implies those vertices (corresponding to the pins) in a given net n_i with smallest and largest x coordinate values. On the other hand, backward search method for a pair of pins starts with the pin with largest x coordinate value (designated as the source vertex) and ends on the pin with smallest x coordinate (treated as the sink vertex).

Now we define the following configurations for conducting the experiments on the

Table 4.5: Comparing Via Count w.r.t γ: between BFS-NB and DFS-NB (in the bracket below)

γ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	Avg. Via
Mode/ Circuit	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	Count
apte	429 (429)	429 (429)	429 (429)	429 (429)	429 (429)	429 (429)	429 (429)	429
hp	453 (455)	453 (455)	453 (455)	453 (455)	453 (455)	453 (455)	453 (455)	454
xerox	1259 (1273)	1235 (1273)	1235 (1273)	1235 (1273)	1235 (1273)	1250 (1293)	1235 (1278)	1259
ami33	1165 (1209)	1181 (1209)	1181 (1209)	1181 (1209)	1181 (1209)	1199 (1199)	1211 1194	1195
ami49	3535 (3547)	3535 (3547)	3535 (3547)	3535 (3547)	3535 (3547)	3629 (3681)	3536 (3546)	3557
n10	235 (227)	235 (227)	235 (227)	235 (227)	235 (227)	234 (227)	234 (227)	231
n30	992 (1006)	992 (1006)	992 (1006)	992 (1006)	992 (1006)	996 (1009)	1001 (1013)	1001
n50	3110 (3295)	3110 (3295)	3110 (3295)	3110 (3295)	3110 (3295)	3120 (3299)	3122 (3280)	3203
n100	7218 (7055)	7218 (7055)	7218 (7055)	7218 (7055)	7218 (7055)	7093 (7075)	7084 (7078)	7121
n200	19290 (19405)	17816 (19405)	19290 (19405)	19290 (19405)	19290 (19405)	19267 (19437)	19222 (19460)	19242
n300	31543 (31596)	31543 (31596)	31543 (31596)	31543 (31596)	31543 (31596)	31665 (31842)	31765 (31889)	31632
Norm. Geo. Mean	**0.996** (1.003)	**0.989** (1.003)	**0.996** (1.003)	**0.996** (1.003)	**0.996** (1.003)	**1.000** (1.009)	**0.997** (1.004)	1.000

specified floorplan instances of each of the benchmark circuits given in Table 4.1:

1. forward search with *No Capacity Scaling* (FCN)
2. forward search with *Hyperbolic Capacity Scaling* (FCH)
3. forward search with *Ladder type Capacity Scaling* (FCL)
4. backward search with *No Capacity Scaling* (BCN)
5. backward search with *Hyperbolic Capacity Scaling* (BCH)
6. backward search with *Ladder type Capacity Scaling* (BCL)

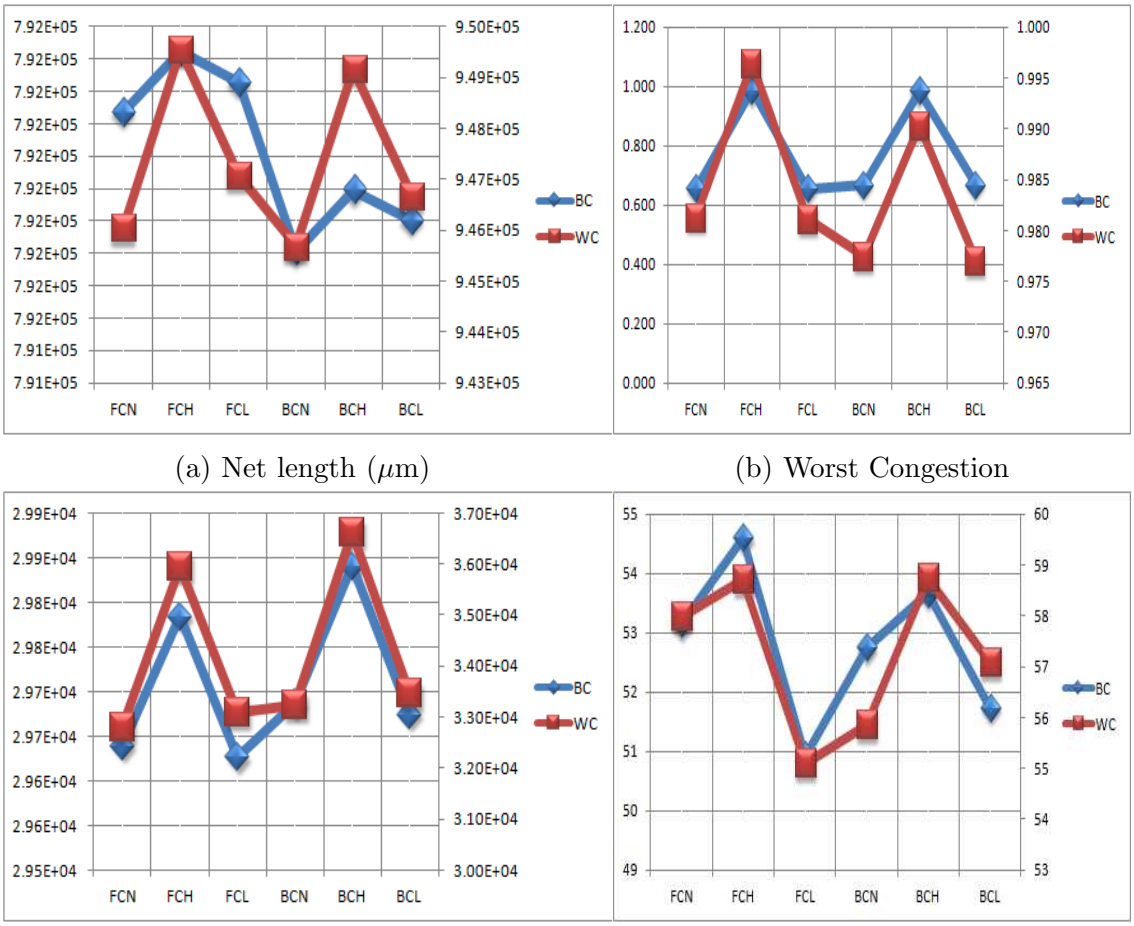
In Figure 4.17, we present a snapshot of the global routing results obtained for $n300$ for all the six run configurations that includes BC and WC different floorplan instances and three different capacity profiles. In these plots, the results for BC (WC) floorplan instance are plotted with respect to the *left (right) axis*. While studying these plots, we notice that the forward (backward) search with hyperbolic scaling FCH (BCH) gives the worst results as compared to the other two configurations $\{FCN, FCL\}$ ($\{BCN, BCL\}$) both in terms of routed net length and via count, both

Table 4.6: Comparing Worst Congestion w.r.t γ: between BFS-NB and DFS-NB (in the bracket below)

γ	0.1	0.2	0.3	0.4	0.5	0.6	0.7
Mode/ Circuit	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)	BFS-NB (DFS-NB)
apte	0.873 (0.881)	0.873 (0.881)	0.873 (0.881)	0.873 (0.881)	0.873 (0.881)	0.873 (0.881)	0.873 (0.881)
hp	0.958 (0.973)	0.958 (0.973)	0.958 (0.973)	0.958 (0.973)	0.958 (0.973)	0.958 (0.973)	0.958 (0.973)
xerox	0.330 (0.675)	0.641 (0.675)	0.641 (0.675)	0.641 (0.675)	0.641 (0.675)	0.659 (0.659)	0.651 (0.772)
ami33	0.948 (0.932)	0.950 (0.932)	0.950 (0.932)	0.950 (0.932)	0.950 (0.932)	0.954 (0.923)	0.954 (0.927)
ami49	0.819 (0.801)	0.819 (0.801)	0.819 (0.801)	0.819 (0.801)	0.819 (0.801)	0.713 (0.711)	0.805 (0.801)
n10	0.766 (0.689)	0.766 (0.689)	0.766 (0.689)	0.766 (0.689)	0.766 (0.689)	0.766 (0.689)	0.766 (0.689)
n30	0.733 (0.770)	0.733 (0.770)	0.733 (0.770)	0.733 (0.770)	0.733 (0.770)	0.735 (0.770)	0.735 (0.770)
n50	0.875 (0.932)	0.875 (0.932)	0.875 (0.932)	0.875 (0.932)	0.875 (0.932)	0.877 (0.940)	0.872 (0.946)
n100	0.909 (0.928)	0.909 (0.928)	0.909 (0.928)	0.909 (0.928)	0.909 (0.928)	0.887 (0.928)	0.883 (0.928)
n200	0.800 (0.812)	0.736 (0.812)	0.800 (0.812)	0.800 (0.812)	0.800 (0.812)	0.799 (0.807)	0.794 (0.782)
n300	0.889 (0.866)	0.889 (0.866)	0.889 (0.866)	0.889 (0.866)	0.889 (0.866)	0.882 (0.913)	0.878 (0.940)
Geo. Mean	**0.783** (0.836)	**0.826** (0.836)	**0.832** (0.836)	**0.832** (0.836)	**0.832** (0.836)	**0.822** (0.829)	**0.829** (0.850)

in BC and WC. This is due to the fact that the hyperbolic profile is the most stringent profile among the other profiles depicted in Figure 4.14. Subsequently, we focus on the corresponding results obtained for the remaining configurations $\{FCN, FCL\}$ ($\{BCN, BCL\}$) for both BC and WC instances. Figure 4.17 (a) shows that FCN (BCN) gives better net length against FCL (BCL) both in BC and WC. Although it reflects a similar trend in the respective via count for the WC topology, FCL (BCL) has better via count as compared to FCN (BCN) in BC (Figure 4.17 (b)). We conduct another set of comparison for net length and via count between FCN and BCN (FCL and BCL) for both BC and WC. Although backward search produces better net length as compared to that in forward search method, it incurs more vias to route a set of nets than its counterpart. This clearly shows that a global routing solution not only depends on the search direction and the capacity profiles, but also on different floorplan instances of the same circuit. We iterate that all the results presented here correspond to 100% routability and restricted to *Under-Congestion* region of Figure 4.6 that ensures no congestion in any routing region in any metal layer. We also notice significant impact on the runtime for each configuration and floorplan instances. We present detailed results for each of the circuits with both BC

and WC floorplan instances and all the configurations in the subsequent tables.



(a) Net length (μm) (b) Worst Congestion

(c) Via Count (d) CPU time (sec)

Figure 4.17: Routing results for BC (left axis) and WC (right axis) floorplan instances of $n300$ vs. different run configurations

In Table 4.7 (4.8), we summarize the netlength obtained for all the configurations for the respective BC (WC) floorplan instances of each of the circuits. We compare them with the respective Steiner length computed by FLUTE [10] presented in the last column in the tables. For a given run configuration, the corresponding netlength is accompanied by the ratio of the routed netlength and Steiner length, e.g., FCN/F in the bracket below it in the respective columns. The corresponding normalized geometric mean values (given the last row of the table) in each column are computed based on these ratios. It is evident from these results on the BC floorplan instances of the circuits that, except for some of the smaller circuits, the length ratio pairs FCN/F and FCL/F in forward search mode (BCN/F and BCL/F in backward search mode)

Table 4.7: Netlength (normalized with Steiner length [10] in bracket) for BC floorplan instances in different run configurations

Circuit	FCN(**μm**) (FCN/F)	FCH(**μm**) (FCH/F)	FCL(**μm**) (FCL/F)	BCN(**μm**) (BCN/F)	BCH(**μm**) (BCH/F)	BCL(**μm**) (BCL/F)	**Steiner** (μm) Length(F)
apte	398137 (1.176)	398411 (1.177)	398137 (1.176)	396034 (1.170)	396309 (1.170)	396034 (1.170)	338628 (1.000)
hp	201997 (1.633)	203060 (1.641)	201997 (1.633)	211537 (1.710)	210086 (1.698)	211537 (1.710)	123716 (1.000)
xerox	716917 (1.132)	717292 (1.132)	716917 (1.132)	710575 (1.122)	710575 (1.122)	710575 (1.122)	633533 (1.000)
ami33	111126 (1.204)	111563 (1.208)	111126 (1.204)	111481 (1.207)	111897 (1.212)	111481 (1.207)	92330 (1.000)
ami49	1925761 (1.197)	2009224 (1.249)	1925761 (1.197)	1926177 (1.197)	2010630 (1.250)	1926177 (1.197)	1608746 (1.000)
n10	19837 (1.193)	19837 (1.193)	19837 (1.193)	18498 (1.113)	18498 (1.113)	18498 (1.113)	16626 (1.000)
n30	59585 (1.207)	59585 (1.207)	59585 (1.207)	58762 (1.190)	58762 (1.190)	58762 (1.190)	49370 (1.000)
n50	151604 (1.213)	152119 (1.217)	151851 (1.215)	150741 (1.206)	151256 (1.210)	150988 (1.208)	125018 (1.000)
n100	251456 (1.186)	252648 (1.191)	251477 (1.186)	250771 (1.182)	251917 (1.188)	250792 (1.182)	212112 (1.000)
n200	429855 (1.128)	430044 (1.129)	429924 (1.128)	428987 (1.126)	429128 (1.126)	429056 (1.126)	381021 (1.000)
n300	792135 (1.133)	792323 (1.134)	792229 (1.133)	791708 (1.133)	791896 (1.133)	791802 (1.133)	699006 (1.000)
Norm. Geo. Mean	1.212	1.219	1.212	**1.205**	1.211	**1.206**	1.000

have little variation and also shows that BCN yields the best net length of a given circuit with respect to the corresponding Steiner length for most of the circuits. This is evident from the geometric mean value of 1.205 as the lowest in BCN mode than any other mode. This implies that the backward search mode without any capacity scaling is more efficient in getting a minimal netlength. However, if we consider capacity scaling across the layers, BCL mode yields the next best value as dictated by the geometric mean value of 1.206. Similar results can also be seen in case of WC floorplan instances where BCN gives the best mean netlength of 1.182 among all, while BCL gives better netlength ratio of 1.183 than BCH, FCL and FCH. All these values are highlighted in the last row of the respective tables for BC and WC instances.

In Table 4.9, we present the via count for all the configurations for each circuit in BC (WC results in brackets). These results clearly point out the consequence of forward and backward search on via count (refer to Figure 4.16). It is also evident that the via count in case of FCH (BCH) is the worst as compared to other two configurations, namely $\{FCN, FCL\}$ ($\{BCN, BCL\}$), during forward (backward) search mode. There is little variation in via count for relatively smaller circuits

Table 4.8: Netlength (normalized with Steiner length [10] in bracket) for WC floorplan instances in different run configurations

Circuit	FCN(**μm**) (FCN/F)	FCH(**μm**) (FCH/F)	FCL(**μm**) (FCL/F)	BCN(**μm**) (BCN/F)	BCH(**μm**) (BCH/F)	BCL(**μm**) (BCL/F)	**Steiner** (μm) Length(F)
apte	450376 (1.155)	450376 (1.155)	450376 (1.155)	438913 (1.126)	439199 (1.127)	438913 (1.126)	389806 (1.000)
hp	232782 (1.606)	232782 (1.606)	232782 (1.606)	230256 (1.588)	230256 (1.588)	230256 (1.588)	144993 (1.000)
xerox	1542730 (1.109)	1590995 (1.143)	1542730 (1.109)	1511243 (1.086)	1558185 (1.120)	1511243 (1.086)	1391401 (1.000)
ami33	120904 (1.151)	120904 (1.151)	120904 (1.151)	118746 (1.131)	118969 (1.133)	118746 (1.131)	105025 (1.000)
ami49	1914369 (1.137)	1914369 (1.137)	1914369 (1.137)	1898528 (1.127)	1898528 (1.127)	1898528 (1.127)	1684114 (1.000)
n10	24526 (1.226)	25116 (1.255)	24526 (1.226)	23708 (1.185)	24298 (1.214)	23708 (1.185)	20012 (1.000)
n30	74743 (1.248)	75105 (1.254)	74838 (1.250)	74152 (1.238)	74514 (1.244)	74247 (1.240)	59879 (1.000)
n50	187971 (1.188)	189752 (1.200)	188476 (1.192)	187197 (1.184)	188994 (1.195)	187702 (1.187)	158173 (1.000)
n100	277305 (1.161)	277951 (1.164)	277584 (1.162)	276545 (1.158)	277251 (1.161)	276824 (1.159)	238841 (1.000)
n200	836136 (1.116)	865590 (1.155)	837493 (1.117)	835535 (1.115)	864329 (1.153)	836928 (1.117)	749479 (1.000)
n300	946039 (1.140)	949547 (1.144)	947076 (1.141)	945689 (1.139)	949166 (1.144)	946676 (1.141)	830035 (1.000)
Norm. Geo. Mean	1.197	1.209	1.198	**1.182**	1.195	**1.183**	1.000

between $\{FCN, FCL\}$ ($\{BCN, BCL\}$) modes as they effectively cater similar routing scenario by using fewer metal layers. However, this variation becomes significant for the larger circuits that use more metal layers where the effect of capacity scaling such as ladder or hyperbolic profile (see Figure 4.14) becomes dominant. The best via counts for each circuit in both BC and WC (in brackets) are mostly obtained in $\{FCN, FCL\}$ modes with similar values, with a normalized geometric mean value of 1.000 for both. The normalized (done with respect to the via count obtained in FCN mode) geometric mean values for each mode for all the circuits are given in the last row of the table for BC instances (WC instances in brackets). These normalized values shows that the hyperbolic profiles in both modes (FCH and BCH) show significant jump (of around 30% with respect to FCN and BCN respectively) in via count due to higher metal pitch as a result of its stringent capacity profiling. The same can also be noticed for individual circuits that use more metal layers to route the nets.

In our congestion analysis, we use a similar method prescribed in *GLARE* [15] to analyze the congestion hotspots in a given floorplan instance of a circuit to estimate its routability, despite that fact that our congestion model (see Equation 4.2) does not allow the routing demand in a routing region in any routing layer to go beyond the

Table 4.9: Via count for BC (WC in bracket) floorplan instances in different run configurations

Circuit	FCN	FCH	FCL	BCN	BCH	BCL
apte	404 (452)	508 (660)	404 (452)	412 (460)	504 (656)	412 (460)
hp	502 (430)	720 (630)	502 (430)	536 (430)	730 (626)	536 (430)
xerox	1190 (1401)	1238 (2261)	1190 (1401)	1220 (1527)	1272 (2369)	1220 (1547)
ami33	1156 (1240)	1500 (1528)	1156 (1240)	1162 (1234)	1530 (1586)	1162 (1234)
ami49	3290 (3629)	4466 (3789)	3290 (3629)	3406 (3877)	4676 (4137)	3406 (3877)
n10	176 (220)	176 (278)	176 (220)	176 (222)	176 (260)	176 (222)
n30	937 (1047)	941 (1100)	937 (1014)	956 (1059)	960 (1108)	956 (1026)
n50	3194 (3252)	3609 (5089)	3184 (3296)	3191 (3402)	3598 (5078)	3181 (3442)
n100	6748 (7742)	7553 (10050)	6748 (7746)	6795 (7846)	7623 (10146)	6799 (7850)
n200	18016 (16905)	18040 (26828)	18008 (17610)	17977 (17253)	17993 (27041)	17969 (17571)
n300	29639 (32814)	29785 (35955)	29627 (33093)	29687 (33235)	29841 (36624)	29675 (33461)
Norm. Geo. Mean	**1.000** (1.077)	1.140 (1.419)	**1.000** (1.080)	1.016 (1.105)	1.153 (1.437)	1.016 (1.106)

specified routing capacity. As discussed in the previous section, some of the edges in the proposed routing graph GSRG become sparse when the congestion (or normalized routing utilization p_e) becomes 1.0 after each net is routed. It thus sets the routing penalty to *infinity* in the corresponding metal layer. The authors in [15] proposed a new parameter called *Average Congestion per Edge* (ACE) for certain percentage of all the (worst) congested global routing edges. This is denoted as $ACE(x\%)$ where x is the percentage value of all the worst congested edges. By computing these parameters, we compute another parameter $wACE4$ which is an weighted average of $ACE(x\%)$ for the specified values of $x = 0.5, 1, 2, 5$. In our case, we set these weights as 1.0 for all the specified values of $ACE(x\%)$ in order to express it merely an average of them. In Table 4.10, we present $wACE4$ values for each circuit for all the configurations in BC and WC (in brackets) to showcase the corresponding congestion scenario with 100% routability of the nets and validate that our global routing framework conforms to the proposed congestion model depicted in Figure 4.6. In these experiments, we

compute $wACE4$ values for each of 8 metal layers and chose the maximum of the respective $wACE4$ values to compare our results. Our study has shown that the maximum values of $wACE4$ are obtained in lower metal layers such as M_1 and M_2.

Table 4.10: Worst congestion ($wACE4$) for BC (WC in bracket) floorplan instances in different run configurations

Circuit	FCN	FCH	FCL	BCN	BCH	BCL
apte	0.901 (0.874)	0.838 (0.986)	0.901 (0.874)	0.945 (0.797)	0.906 (0.875)	0.945 (0.797)
hp	0.991 (0.991)	0.987 (0.814)	0.991 (0.991)	0.987 (0.991)	0.985 (0.802)	0.987 (0.991)
xerox	0.623 (0.994)	0.614 (0.937)	0.623 (0.994)	0.623 (0.958)	0.609 (0.937)	0.623 (0.958)
ami33	0.967 (0.939)	0.687 (0.994)	0.967 (0.939)	0.971 (0.912)	0.692 (0.810)	0.971 (0.912)
ami49	0.975 (0.713)	0.993 (0.721)	0.975 (0.713)	0.975 (0.711)	0.998 (0.707)	0.975 (0.711)
n10	0.643 (0.962)	0.643 (0.950)	0.643 (0.962)	0.643 (0.962)	0.767 (0.950)	0.643 (0.962)
n30	0.658 (0.955)	0.983 (0.967)	0.658 (0.955)	0.657 (0.955)	0.979 (0.950)	0.657 (0.955)
n50	0.818 (0.840)	0.663 (0.825)	0.821 (0.849)	0.816 (0.865)	0.634 (0.835)	0.818 (0.873)
n100	0.990 (0.919)	0.977 (0.997)	0.990 (0.919)	0.987 (0.928)	0.988 (0.998)	0.987 (0.928)
n200	0.489 (0.875)	0.827 (0.679)	0.489 (0.855)	0.483 (0.892)	0.815 (0.646)	0.483 (0.833)
n300	0.656 (0.981)	0.986 (0.996)	0.656 (0.981)	0.668 (0.977)	0.984 (0.990)	0.668 (0.977)
Norm. Geo. Mean	**1.000** (1.178)	1.065 (1.153)	**1.000** (1.177)	1.004 (1.167)	1.085 (1.110)	1.004 (1.161)

In Table 4.11, we report runtime in seconds for all the circuits for the said run configurations. These results correspond to both BC and WC (given in brackets) floorplan instances. As we can see that the best (as highlighted) runtime in the context of BC and WC (in brackets) floorplan instances for a given circuit is given by either FCL or BCL for most of the cases.

4.6 Chapter Summary

In this chapter, we present the first early global routing method STAIRoute using monotone staircases as the routing regions defined in a floorplan, by recursive floorplan bipartitioning methods presented in Chapter 3. Unlike the existing post-

Table 4.11: CPU time (sec) for BC (WC in bracket) floorplan instances in different run configurations

Circuit	*FCN*	*FCH*	*FCL*	*BCN*	*BCH*	*BCL*
apte	0.114 (0.112)	0.109 (0.106)	0.103 (0.102)	0.108 (0.103)	0.109 (0.104)	0.113 (0.103)
hp	0.115 (0.114)	0.107 (0.107)	0.105 (0.100)	0.107 (0.107)	0.102 (0.102)	0.103 (0.101)
xerox	0.126 (0.140)	0.124 (0.125)	0.122 (0.116)	0.122 (0.130)	0.121 (0.125)	0.121 (0.120)
ami33	0.172 (0.186)	0.167 (0.178)	0.151 (0.158)	0.161 (0.158)	0.159 (0.166)	0.156 (0.170)
ami49	0.464 (0.483)	0.453 (0.465)	0.438 (0.446)	0.441 (0.463)	0.458 (0.465)	0.442 (0.460)
n10	0.122 (0.109)	0.099 (0.101)	0.101 (0.105)	0.101 (0.103)	0.101 (0.105)	0.102 (0.102)
n30	0.181 (0.162)	0.164 (0.161)	0.166 (0.153)	0.164 (0.163)	0.160 (0.164)	0.164 (0.157)
n50	0.423 (0.449)	0.410 (0.437)	0.398 (0.440)	0.414 (0.444)	0.399 (0.441)	0.396 (0.432)
n100	2.48 (2.076)	2.408 (2.096)	2.412 (2.064)	2.399 (2.073)	2.445 (2.070)	2.386 (2.054)
n200	18.342 (21.472)	18.741 (20.885)	17.718 (20.909)	18.201 (21.452)	18.175 (21.315)	17.533 (20.625)
n300	53.151 (57.993)	54.596 (58.719)	50.929 (55.090)	52.739 (55.856)	53.656 (58.742)	51.73 (57.069)
Norm. Geo Mean	1.000 (1.009)	0.955 (0.971)	**0.925** (**0.934**)	0.944 (0.962)	0.940 (0.965)	0.933 (0.947)

placement global routers, the proposed method immediately follows floorplanning in the physical design flow and works only on the floorplan level information, thus requiring no detailed placement of standard cells. Hence, the routing of the nets connecting the standard cells are not in the scope of this work. The proposed routing model supports both unreserved or reserved layer model for routing the nets through multiple metal layers.

In this work, we proposed a new MST based multi-terminal net decomposition technique, monotone staircase pattern routing between any two terminal net segment. Using illustrations, we show that this method is similar to the existing approaches for routing blockage aware Steiner topology generation used in the post-placement global routers. Unlike the overflow based congestion hotspot analysis used in most of the existing global routers, the proposed early global routing method adopts the definition of congestion as the ratio of routing demand and capacity in any routing region (edge). The proposed congestion model ensures the normalized routing utilization

(congestion) in any edge does not exceed beyond 100% in any routing layer. We adopted a new congestion analysis approach, presented in [15], for computing the worst congestion values in certain edges of the proposed early global routing graph.

Using the results of BFS and DFS based bipartitioning approaches, the early global routing results obtained by STAIRoute show slightly improved routing metrics for BFS over DFS, for different γ values. The experiments also show that 100% routing completion is feasible without any over congestion ($congestion \not> 100\%$) even for different capacity profiles. These profiles include the recent trend of metal pitch/width variation across the routing layers prescribed by VDSM fabrication process nodes. Additionally, two terminal net (segment) routing employing different search directions depicts potential improvement in routed wirelength and via count.

Chapter 5

Unconstrained Via Minimization in Early Global Routing

5.1 Introduction

In the existing physical design (PD) flow depicted in Figure 1.9 (a), the global routing stage comprises of two key phases: (i) first a $2D$ (also known as planar) routing solution of the nets is obtained for two routing layers only, and (ii) next a congestion aware layer assignment of the segments (subnets) of the nets is done for more than two layers. The latter is commonly known as $3D$ routing solution. During this layer assignment process, a minimal via routing solution is said to be obtained without changing the topology of the nets on the basis of the $2D$ routing solution. If this process does not yield a congestion free routing solution or a minimized congestion scenario to some predefined acceptable values, the violating nets are ripped up and rerouted iteratively. On the other hand, there is no guarantee that the number of vias used to complete the layer assignment process of the nets is minimal even if a routing solution does not result in any violation due to over congestion in certain edges in the grid graph. This process is called *constrained via minimization* (CVM).

Another well known via minimization method that is hardly in practice is called *unconstrained via minimization* (UVM). In this method, an emphasis is given on a minimal via routing path of a net while ensuring 100% routability, lower congestion in a routing layer and a minimal routed wirelength. In this method, a planar routing solution like in CVM becomes redundant. Both CVM and UVM are NP-hard problems [17] and UVM is known to be much harder than CVM [17]. In this chapter, we

address the UVM problem in the proposed early global routing framework STAIRoute presented in Chapter 4. This is done by suitably identifying a set of monotone staircases in a given floorplan with minimal number of bends in it, by a recursive floorplan bipartitioning framework similar to that presented in Algorithm 2.

5.2 Via Minimization in Global Routing

In order to reduce the number of vias in a design during the recent global routing methods, a set of routing patterns with fixed number of bends, such as L with one bend and Z with two bends, have become increasingly popular. For a minimum of one layer change (say M_1 and M_2), these patterns use one and two vias respectively at the corresponding bends adjoining consecutive routed (vertical/horizontal) segments [19, 20]. These pattern routing methods are much faster than the maze routing methods for obtaining a feasible routing path of a net through a set of routing regions. Another pattern, known as 3-bend routing [27], has been proposed in order or reduce the number of vias and mitigate congestion hotspots. In some congestion scenario, this kind of patterns yield detour out of the net bounding box leading to increased wirelength for reducing congestion and via count.

An extension of this pattern routing using more than two bends was subsequently proposed in [19], with a similar runtime overhead as that for Z but much faster than maze routing. This pattern routing method is known as *monotone staircase pattern routing* and is more flexible in finding a routing path through a region with considerable number of routing blockages over one/two bend (L/Z) routes. It is to be noted that this multi-bend monotonic pattern routing is a generalization of L and Z pattern routing. Notably, like L and Z patterns, a monotonic routing path is also confined within the bounding box of the net pins. This implies that a net when routed using pattern routing such as L, Z or even monotonic patterns yields the routed (planar) wirelength equal to the half perimeter wirelength (HPWL) of the bounding box of the net. However, the number of vias required along a monotonic routing path increases proportionally with the number of bends in it, with at least one more than Z pattern. This also depends on how many layer changes take place at each bend in this pattern. Therefore, a careful trade off has to be made between the flexibility in routing completion and the number of bends while using the monotone staircase patterns for routing the nets.

5.3 A BFS based Greedy Method

The proposed early global routing method STARoute in the Chapter 4 identifies the routing paths through a set of contiguous rectilinear regions bounded by the block boundaries and their intersection points (also known as T-Junctions). The number of these T-junctions in any floorplan containing n blocks is $2n-2$ [105] (also Lemma 6), leading to a total of $2n-2$ bends in the entire floorplan. Figure 5.1 illustrates four different orientation of a T-junction. Each orientation of a T-junction can either have a upward or a downward bend depending on the corresponding ms-cut on BAG as an MIS or MDS cut (see Chapter 3). Therefore, a total 8^{2n-2} combinations of bends are possible for the entire floorplan of the design. However, the recursive bipartitioning framework presented in Chapter 3 identifies only one combination of $2n-2$ bends out of that exponentially large number of possible combinations. These bends collectively belong to all the monotone staircase regions obtained recursively for the entire floorplan. Depending on the bipartitioning results, we obtain different number of bends for each of the monotone staircases obtained by those floorplan bipartitioning methods, although we did not give any emphasis on the bends for individual monotone staircase region. This chapter presents a new bipartitioning framework that emphasizes on these bends in addition to the objectives defined in Chapter 3.

5.3.1 Problem Definition

Since the primary focus in this work is to realize an early global routing framework that employs UVM for early via minimization, we intend to identify a set of monotone staircases in a given floorplan that will facilitate multi-bend pattern routing including L and Z patterns through a set of metal layers; as the special cases of monotone staircase pattern routing having one and two bend along the routing path. In this section, we present a floorplan bipartitioning method using breadth first traversal on the block adjacency graph of a given floorplan topology [114]. This method identifies a set of minimal bend monotone staircase routing regions in a given floorplan that will identify a minimal via routing path for the given net through multiple metal layers. We have already discussed in Chapter 3 that an area balanced floorplaning bipartitioning is a multi-objective optimization problem. We summarize the objectives of this floorplan optimization problem, by augmenting the problem defined in Section

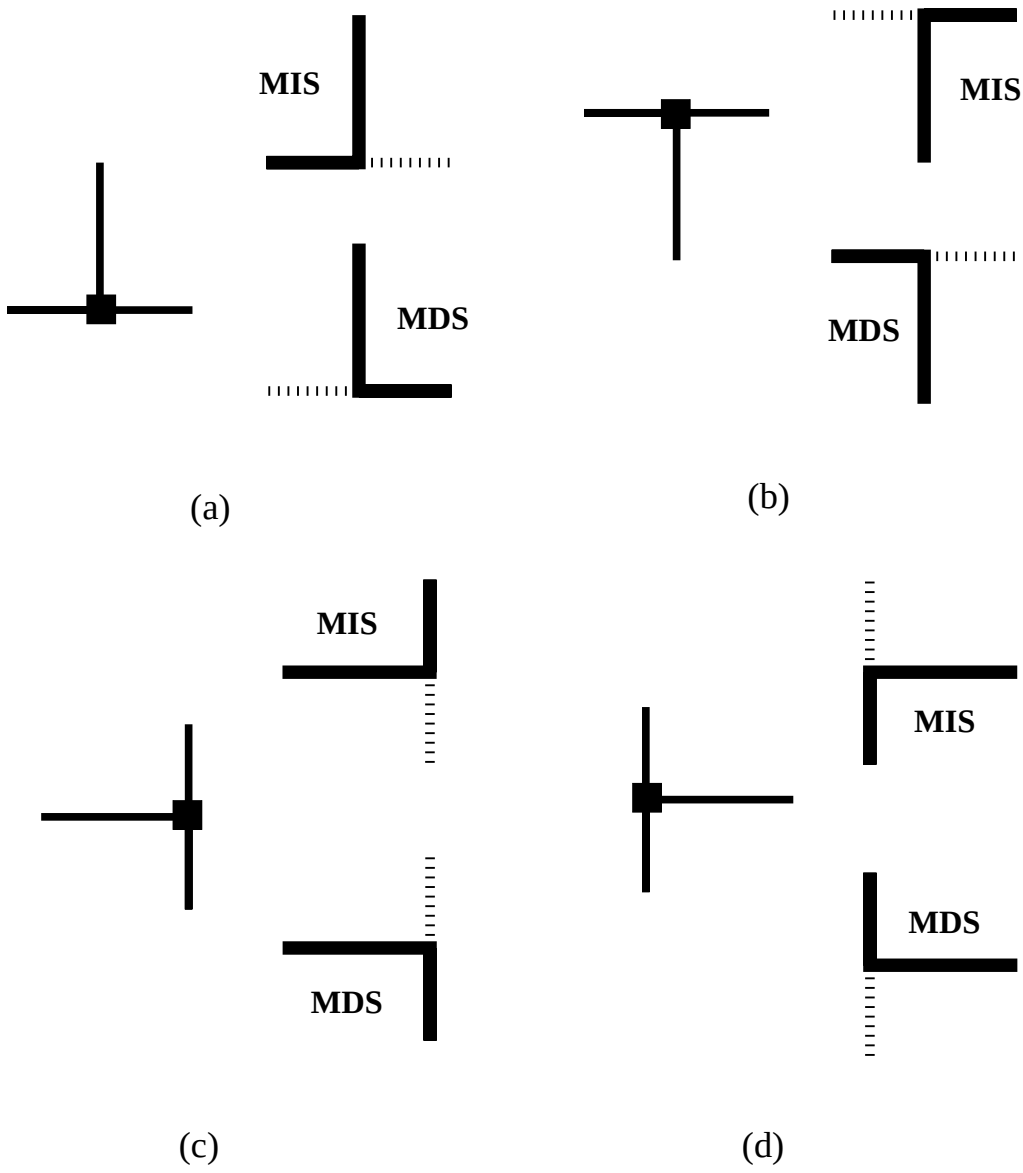


Figure 5.1: Enumerating different orientations of a T-junction (■) and the possible bends (in bold line) pertaining to an MIS/MDS staircases

3.4, as below:

1. balance ratio $balr = \min(A_l,A_r)/\max(A_l,A_r)$ be maximized
2. the number of cut nets (k_c) be minimized, *and*
3. the number of bends (z) in the monotone staircase be minimized

In addition to the trade-off parameter $\gamma \in [0, 1]$ defined in Section 3.4, we introduce another trade-off parameter $\beta \in [0, 1]$. Using these two parameters, we present a

modified *Gain* function presented in Equation 3.1 in order to incorporate this new objective for identifying a minimal bend monotone staircase as below:

$$Gain = \gamma.balr + (1 - \gamma - \beta)(1 - k_c/k) + \beta(1 - z/z_{max}) \tag{5.1}$$

Here z_{max} is the maximum number of possible bends if the constituent rectilinear segments in the corresponding monotone staircase had alternating orientation (vertical or horizontal) and computed as one fewer than the number of such segments in the staircase region. Notably, this equation reduces to Equation 3.1 when the value of β is 0.0. Careful selection of a (γ, β) pair among a range of values yields an optimal balance among the said objectives. For a given (γ, β) value, the maximum *Gain* is obtained by computing its values for the all the bipartitions obtained iteratively at each level of the bipartition hierarchy (refer to Figure 5.3) and taking the maximum of them. An optimal monotone staircase cut on the BAG is the one that pertains to the cut that yields maximum *Gain* for the given (γ, β) pair.

5.3.2 Illustration on Minimal Bend Monotone Staircases

In this section, we study how the bends in a monotone staircase pattern impact the number of vias due to the layer assignment of the nets on multiple metal layers. These vias make the electrical interconnections between the net segments routed in different metal layers. In the reserved layer model, each layer supports a preferred routing direction such as vertical or horizontal routing. For example, metal layer M_1 allows routes of the horizontal net segments, while M_2 allows only vertical routing. During layout implementation, if the routing tools incur too many vias to complete the routing of the nets, it reduces the reliability of a design due to significant failure probability of the vias and also increases the power consumption due to higher contribution of resistance due to these vias. Moreover, a design using too many routing (metal) layers for better routing completion and improved timing performance increases the layout manufacturing cost. During design implementation process, the effort is given to route the nets through fewer routing layers as possible and also incurring fewer via count leading to less impact on the timing performance of the design.

Before discussing the proposed method for minimal bend monotone staircase bipartitioning method, we study how the number of bends in a monotone staircase

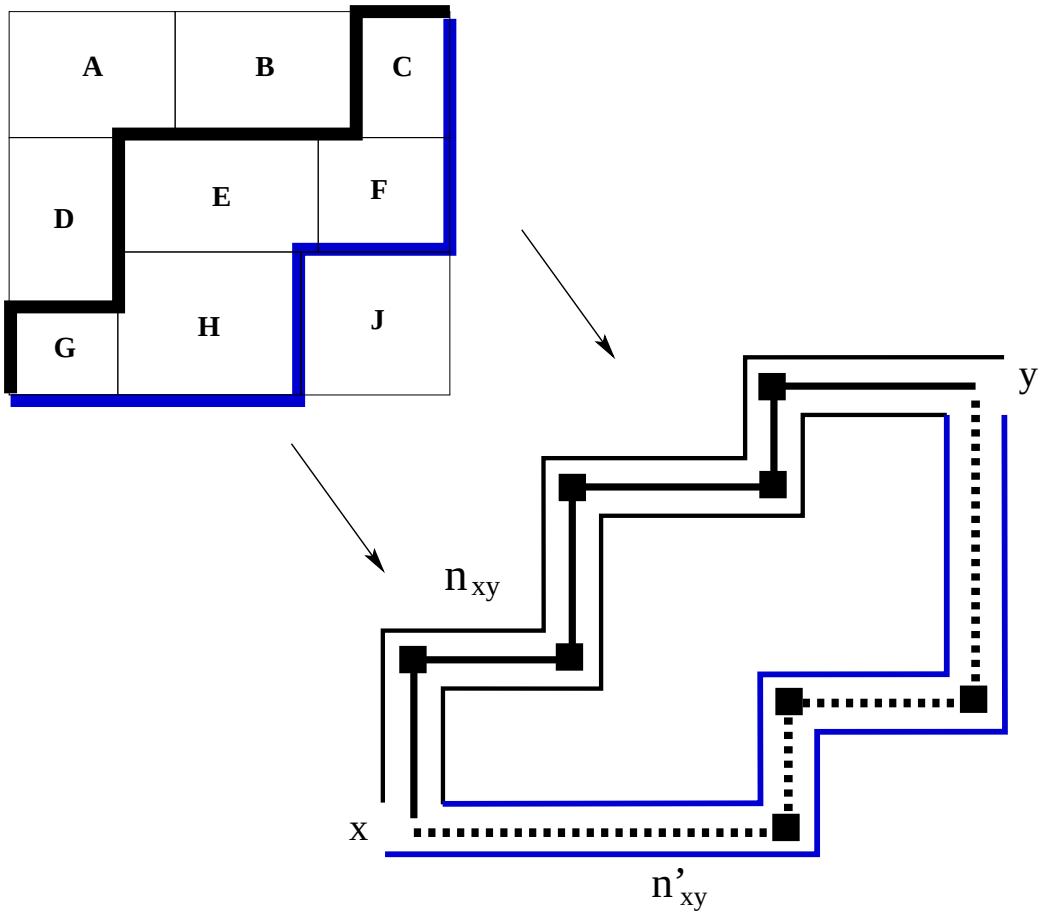


Figure 5.2: Impact of bends along a monotone staircase routing path on the number of vias (marked as ■) obtained during global routing

routing region impacts the number of vias when a net is routed through it in a small example presented in Figure 5.2. In this example, we assume reserved layer model that facilitates the assignment of the net segments of a net on various metal layers based on their vertical/horizontal orientation and restrict to two metal layers only. Although, it can further be extended to any number of metal layer pairs supported by the IC fabrication processes. We consider two different routing instances between point x and y of a net segment n, denoted as n_{xy} and n'_{xy} respectively, through two different monotone staircase routing paths with different bend counts as depicted. It shows that the routing path n_{xy} takes five vias for the interconnection of the alternating net segments, while n'_x requires only three vias. From this example, we infer that a monotone staircase with minimal bends may potentially reduce the number of vias at the global routing stage during layer assignment and thus motivates this work.

Now, we study how to identify a monotone staircase with minimal number of bends in it using a recursive floorplan bipartitioning method for a given floorplan, while no other earlier maxflow based works [8, 102, 103] and the BFS/DFS based bipartitioning methods proposed in Chapter 3 considered this objective in this multi-objective problem.

Our study on Figure 5.3 shows that a floorplan bipartitioning method can address the problem of maximizing the area of each partition while identifying a monotone staircase with minimal number of bends. We can also extend this study in the context of minimal net cut, but for the sake of simplicity we restrict it to area balance and bend minimization only. As we can see that each bipartition of the floorplan in illustrated in Figure 5.3 (a) gives a minimum number of bends ($z = 3$), but comes with a poor area balance. The area balance between the partitions keeps on improving through Figure 5.3 (b)-(e) with varying number of bends in the resulting monotone staircase, while it declines for the instances shown in Figures 5.3 (f)-(h). For this floorplan topology, the best possible area balance may be achieved in the case of Figure 5.3 (e), but at the cost of higher bend count ($z = 6$). Therefore, we need to make a suitable trade-off between area balance and bend count. Among all the explored solutions illustrated in this example, the bipartition instance with $z = 4$ in Figure 5.3 (d) can be a good choice among all the other instances. We must also consider the corresponding net cut for each of the bipartition instances depending on the trade-off parameter values (γ, β).

Lemma 9 *Given a floorplan with n blocks, the number of bends in a monotone staircase is $O(n)$.*

Proof The number of bends in the resulting monotone staircase can be at most one fewer than the number of cut edges, due to alternate orientation of the contiguous cut edges in BAG. Thus the number of cut edges, being a subset of E_b, that constitutes a monotone staircase, is $O(n)$. □

5.3.3 The Algorithm

Here, we present the pseudo-code for the proposed monotone staircase bipartitioning with minimal bends, namely *GenMSCut-Bend*, in Algorithm 5. The key differences between *GenMSCut-Bend* and Algorithm 1 are:

1. bend minimization incorporated as an additional objective, and

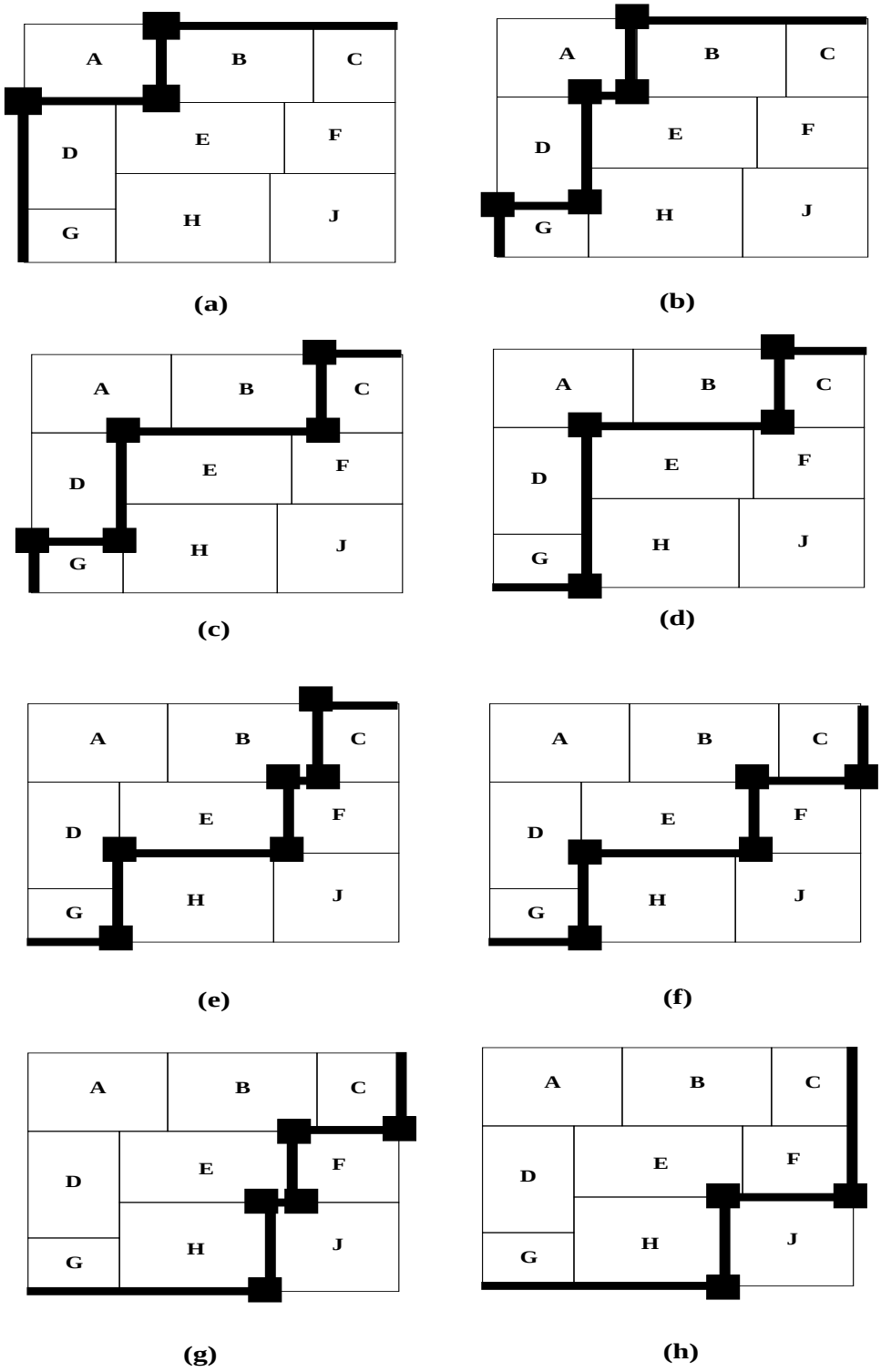


Figure 5.3: The number of bends (■) (z) for a sequence of monotone staircases in a floorplan: (a) 3, (b) 5, (c) 5, (d) 4, (e) 6, (f) 5, (g) 5 and (g) 3

2. no restriction on convergence due to predefined area bounds; instead all $n-1$ bipartitions are explored in order to find the best solution.

In this algorithm, similar to Algorithm 1, the inputs are the block adjacency graph (BAG) G_b for a given (sub)floorplan $F(B, N)$ for a set of blocks B and nets N, at any level of the bipartition hierarchy. In addition to these, we also provide (γ, β) values and the type of the bipartition i.e. area or number balanced.

5.3.3.1 The Recursive Framework

The recursive procedure for obtaining a set of minimal bend monotone staircase for a given floorplan $F(B, N)$ with a set of blocks B and nets N is presented in Algorithm 6. This procedure is an improvement over the recursive framework (Algorithm 2) presented in Section 3.4, i.e., the omission of the convergence parameter ϵ. The rest of the parameters carry the same meaning as in Section 3.4. In this work, we focus mainly on area balanced bipartitioning by setting *baltype* to 1 at each level of the bipartition hierarchy, since the number balanced bipartitioning method is a special case of it and its results on the given benchmarks (presented in Chapter 3) are not superior due to nonuniform area distribution of the blocks.

Theorem 3 *Given a floorplan with n blocks and k nets, GenMSCTree-Bend takes $O((n^2+nk)\log n)$ time to generate a hierarchy of minimal bend monotone staircases.*

Proof Since the BAG G_b for a given floorplan is a planar graph, its construction takes $O(n)$ time. By Lemma 9, each while loop in Algorithm 5 takes $O(n)$ for identifying the bends and $O(k)$ for net bipartition (see Chapter 3). Hence, each call to GenMSCut-Bend takes $O((n + n^2 + nk)\log n)$, i.e., $O(n^2 + nk)$. Therefore, for the bipartition hierarchy with $O(\log n)$ levels, GenMSCTree-Bend takes $O((n^2 + nk)\log n)$ time to identify all the monotone staircases with minimal bends in the entire floorplan. □

5.4 A Randomized Wave Propagation Method

We have seen in Section 3.4 that the number of monotone staircases obtained at any level of bipartition hierarchy is $n-1$, for a floorplan containing n blocks, as per Lemma 4. But the number of all possible monotone staircases in a given floorplan topology is exponentially large, and the problem of finding the optimal area (number)

Algorithm 5 GenMSCut-Bend

Inputs: G_b, N, γ, β, *baltype*

Outputs: An ms-cut with maximal area balance, minimal net cut and number of bends, yielding maximum *Gain* value (see Equation 5.1) for a given (γ, β) pair

```
Define a Queue, Q, of size 2n, S = source(G_b)
Initialize left partition, L = ∅, Wt(L) = 0, λ = ∅
level_no = 0, Enqueue(S), S.level = level_no
Enqueue(∅)
while (NOT_EMPTY(Q)) do
  v_i = Dequeue(Q)
  if (v_i ≠ ∅) then
    L = L ∪ {v_i}; Level(v_i) = level_no
    Wt(L) ← Wt(L) + Wt(v_i)
    for (v_j ∈ adj(v_i)) do
      if (IsValid_Monotone_Staircase = TRUE) then
        /* IsValid_Monotone_Staircase implies Lemma 1 */
        Enqueue(v_j)
      end if
    end for
    /* here B_l (B_r) corresponds to L (V_b − L)*/
    findPartition(B_l, B_r, N).
    /* returns N_l(N_r), N_c, Z (Z_max) being the set of left (right), cut nets and bends
    (maximum possible bends) respectively */
    Calculate Gain (see Equation 5.1) using B_l, B_r, k_c = |N_c|, k = |N|, z = |Z|
    and z_max = |Z_max| for a given (γ, β)
    C_p = ⟨Gain, B_l, B_r, N_l, N_r, N_c, Z, Z_max⟩
    λ ← λ ⋃ {C_p}
  else
    Increment level_no
    Enqueue(∅)
  end if
end while
Return an optimal ms-cut C_max ∈ λ with maximum Gain
```

balanced monotone staircase cut on the corresponding BAG is known to be an NP-Hard problem [8, 103]. Given a floorplan topology $F(B, N)$ for a set blocks B and nets N, it is intuitive that a monotone staircase cut, represented as the left partition L and the right partition R, on the BAG can be represented as a subset of B ($L \cup R = B$ and $L \subset B$). Therefore, the set of all possible monotone staircases $\mathcal{S}$ is a subset of the power set of B, i.e., $\mathcal{S} \subseteq$ power(B). In other words, the elements in power(B)

Algorithm 6 GenMSCTree-Bend

Inputs: B,N,F,*stype*,γ, β,*baltype*

Outputs: The *MSC tree*, with increasing (decreasing) ms-cuts MIS (MDS) at alternate level

if ($Root\_node \parallel TreeLevel\%2 = 0$) **then**
  $stype = 1$
else
  $stype = 0$
end if
G_b = ConstructBAG($B, F, stype$)
Node.cut = GenMSCut-Bend(G_b, N, γ, β, *baltype*)
Node.Level = $TreeLevel$; increment $TreeLevel$
if ($|B_l| \geq 2$) **then**
  Node.left = GenMSCTree-Bend($B_l, N_l, F_l, stype, \gamma, \beta$,*baltype*)
end if
if ($|B_r| \geq 2$) **then**
  Node.right = GenMSCTree-Bend($B_r, N_r, F_r, stype, \gamma, \beta$,*baltype*)
end if
return Node.

that represent all possible monotone staircases $\mathcal{S}$ must obey the *monotone staircase property* (see Lemma 1 in Chapter 3). So far, several heuristic approaches using BFS/DFS method in floorplan bipartitioning have been proposed including those in Sections 3.4 and 5.3 in order to identify a sequence of $n-1$ monotone staircases in a given floorplan at each of level of the bipartition hierarchy.

It is to be noted that the set of all possible monotone staircases $\mathcal{S}$ for a given floorplan is a partially ordered set. The corresponding *Hasse Diagram* with all the elements in this set depicted in Figure 5.4 can be constructed accordingly. The hasse diagram has two special nodes, one at the top named as *START* and another at the bottom called **STOP**. The number of nodes in this diagram, corresponding to all the possible staircases for a given floorplan topology, grows exponentially with the number of blocks in the floorplan. Each node represents a distinct monotone staircase while each edge represents a possible transition from one distinct monotone staircase to another. It is also to be noted that a node can belong to at least one sequence of staircases. For example $\{1, 2\}$ in Figure 5.4 (b) belongs to two different partially disjoint sequence from START node to STOP node. It is also evident from this diagram that a path from the top most node to the bottom most node (*START*

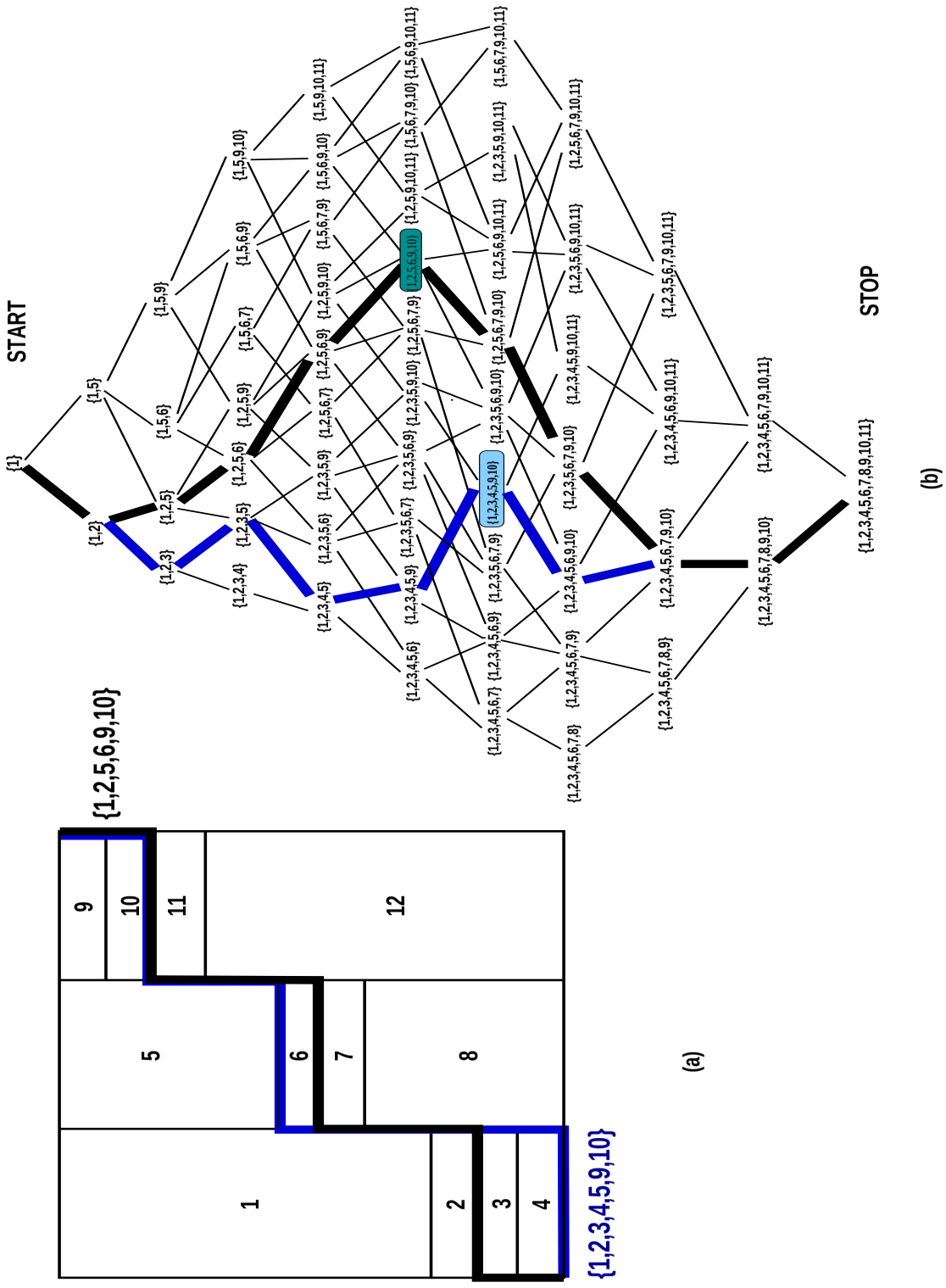


Figure 5.4: (a) A floorplan having 12 blocks with a near optimal staircase $\{1, 2, 5, 6, 9, 10\}$, and (b) a Hasse Diagram containing exponentially large number of sequences (paths) of monotone staircases: one path (blue) containing the optimal monotone staircase $\{1, 2, 3, 4, 5, 9, 10\}$ and the other (in black) containing a near optimal monotone staircase $\{1, 2, 5, 6, 9, 10\}$

$\rightsquigarrow$ *STOP*) consists of several such edges and adheres to a particular sequence. The length of any such sequence is always $n-1$ as per Lemma 4. However, the number of such paths (sequences) grows exponentially with the number of blocks n in a floorplan instance F, but there is no necessity that these paths are fully disjoint. Notably, the sequences also differ from one floorplan instance to another for the same set of blocks B, and hence lead to a completely different hasse diagram. Unlike in Chapter 3, the proposed bipartitioning method in this section finds an optimal solution from a collection of minimal bend monotone staircases belonging to more than one sequence. The corresponding bipartition implies an optimal trade-off among the constituent objectives given in the previous section as: (a) maximizing area of each partition, (b) minimizing the number of bends in the corresponding monotone staircase, and (c) the number of nets cut by the bipartition.

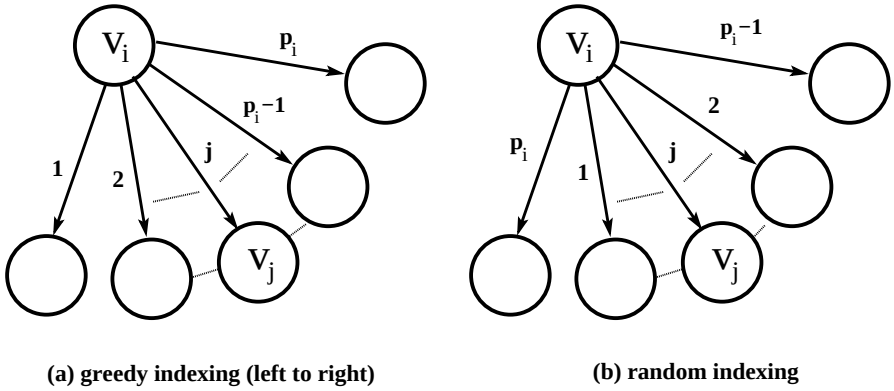


Figure 5.5: Neighbor Indexing: (a) greedy (left to right), and (b) random

In this section, we study how the proposed randomized technique picks the neighbors of a vertex v_i in BAG already included in L partition, during each step of breadth first traversal. This leads to different sequences of monotone staircases during different trials. Here, the neighbors of v_i are those adjacent vertices which are not yet explored during BFS traversal. As compared to the BFS based greedy method in Section 3.4 (also in Section 5.3) where the neighbors are ordered from left to right with the increasing indices (smaller to larger values of lower left x coordinates of the blocks) depicted in Figure 5.5 (a), the neighbors of a vertex with out-degree p are indexed randomly with an index $j \in [1, p]$ (see Figure 5.5 (b)). Therefore, searching for a neighbor with any index j does not have any bias. This implies all the neighbors are equally probable to be picked as the next vertex. As a results, this selection may potentially yield a completely new sequence that may not have been possible to be

discovered during the earlier greedy methods.

Lemma 10 *For a given vertex v_i with out-degree p in G_b, the expected time $E[t_{adj}]$ to search its adjacency (neighbor) list in order to identify one or more potential monotone staircases is $(p+1)/2$.*

Proof Since all the vertices in the adjacency list are equally probable to be picked, with a probability of $1/p$, the expected runtime to search a particular adjacent vertex v_j with random indexing $j \in [1, p]$ is:
$E[t_{adj}] = \sum_{j=1}^{p} (1/p).j = (p+1)/2$ □

Alike the BFS method in Sections 3.4, the best case scenario occurs when all the p edges emanating from v_i obey the monotone staircase property (Lemma 1), resulting in p distinct monotone staircases. The worst case scenario occurs when the number of edges that obey Lemma 1 is 1, resulting in only one monotone staircase. The following lemma gives a measure on the average number of monotone staircases that can be obtained for each vertex.

Lemma 11 *For a given vertex v_i in G_b with out-degree p, $O(p)$ distinct monotone staircases can be identified while obeying Lemma 1.*

Proof Since, each edge in the BAG has a probability of 1/2 in obeying Lemma 1, the average number of distinct monotone staircases
$= 1/p(1 + 2 + ... + (p-1) + p)$
$= (p+1)/2$ □

5.4.1 Illustration

We take an example floorplan with $n = 12$ and $|\mathcal{S}| = 56$ overlaid with two different monotone staircases in Figure 5.4 (a). The corresponding *Hasse Diagram* [115, 116] in Figure 5.4 (b) illustrates the respective sequences of monotone staircases marked by *blue* and *black* lines. During a trial, if the blue path does not contain an optimal monotone staircase, it demands another path to be explored in a hope to identify an optimal one. In this case, we can not apply a brute force method to search the exponentially large number solution space in order to identify an optimal solution. However, a randomized selection of the adjacent vertices of a vertex in the BAG G_b during the breadth first traversal (unlike Algorithm 1 or 5) can be useful. Careful

study of Figure 5.4 (b) shows that randomization at the suitable node, say in this case $\{1, 2\}$, by selecting the neighbor block 3 randomly instead of 5 may guide to a different sequence that contain an optimal solution $S_{opt} = \{1, 2, 3, 4, 5, 9, 10\}$.

In summary, if several such randomized selections are applied at each subsequent node during each trial, this method may yield an optimal solution; or at least provides a better scope of selecting the best solution along a specific path ($START \rightsquigarrow STOP$). A number of such trials may be exercised in order to explore different sequences and thus potentially improve the quality of the solution in terms of the optimality of the said objectives. In order to contain the run time within the same bound as in Algorithm 5, we can not afford to have very large number of trials; instead we restrict the number of trials to a reasonably small value and use random seeds during each trial. The best monotone staircase is identified as the one, with the maximum *Gain* value, among all the staircases explored along different paths; we consider it as an optimal solution. And example in Figure 5.6 showcases how we obtain different sequences during three different trials of random neighbor selection.

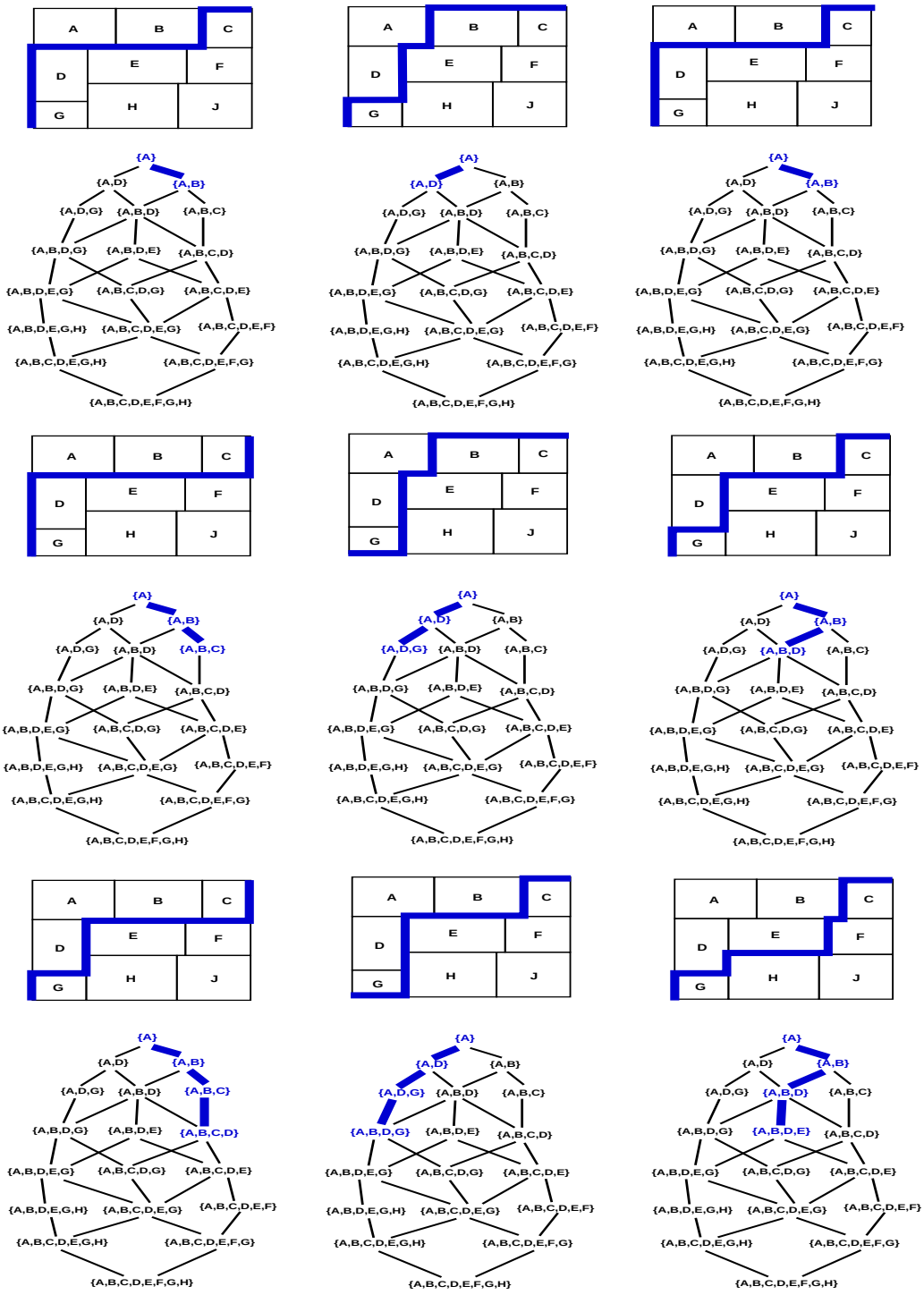
5.4.2 The Algorithm

The pseudo-code for the proposed floorplan bipartitioning method *GenMSCut-Rand* random neighbor search technique is presented in Algorithm 7 [117].

Lemma 12 *At any given node of the bipartition hierarchy, the proposed randomized method GenMSCut-Rand takes $O(n^2 + nk)$ time for identifying an optimal minimal bend monotone staircase in a floorplan containing n blocks and k nets.*

Proof Since the BAG G_b of a floorplan is planar, the number of edges $|E_b|$ in it is $O(n)$ and $E_b = \sum_{i=1}^{n} \mathrm{p_i}$, p_i being the out-degree of a vertex v_i. Therefore, $O(n)$ time is needed for searching distinct monotone staircases in the floorplan while exploring all the vertices in G_b except the sink vertex. Moreover, we use a fixed number of trials intended to identify different sequences of monotone staircases. Also the net partitioning procedure takes $O(k)$, while finding the number of bends accounts for $O(n)$ time (see Lemma 9). Thus, the overall time taken by GenMSCut-Rand is $O(n(n + k))$, i.e., $O(n^2 + nk)$. □

Note that Algorithm 7 has the same $O(n^2 + nk)$ time complexity as the greedy method presented in Algorithm 5, but only a constant times higher due to multiple



(a) 1st trial **(b) 2nd trial** **(c) 3rd trial**

Figure 5.6: Sequences of monotone staircases during different trials of randomized bipartition

Algorithm 7 GenMSCut-Rand

Inputs: G_b, N, γ, β, *baltype*

Outputs: A ms-cut with maximal area balance, and minimal net cut and number of bends with maximum *Gain* value (see Equation 3.1)

Define a Queue Q of size $2n$; S = source(G_b)
$\lambda = \varnothing$; $r = 0$
while ($r < 3$) **do**
  Initialize left partition, $L = \varnothing$, $Wt(L) = 0$
  $level\_no = 0$, Enqueue(S), S.level = $level\_no$
  Enqueue($\varnothing$)
  while (NOT_EMPTY(Q)) **do**
    v_i = Dequeue(Q)
    $V_{list} = \varnothing$
    if ($v_i \neq \varnothing$) **then**
      $V_{list} \leftarrow \{v_i\}$
    else
      while There exists at least one cut edge in E_b for the vertex front V_{list} **do**
        Generate a random seed based on trial number r
        Randomly choose a cut edge (v_i, v_j), such that $v_i \in V_{list}$ and $v_j \notin V_{list}$
        if ($IsValid\_Monotone\_Staircase$ = TRUE) **then**
          /* $IsValid\_Monotone\_Staircase$ implies Lemma 1 */
          $L = L \cup \{v_i\}$; Level(v_i) = $level\_no$
          $Wt(L) \leftarrow Wt(L) + Wt(v_i)$
          Enqueue(v_j)
          findPartition(B_l, B_r, N)
          {returns $N_l(N_r), N_c, Z(Z_{max})$: set of left(right) nets, cut nets and bends (maximum possible bends) respectively}
          Calculate *Gain* (see Equation 5.1) using $B_l, B_r, k_c = |N_c|, k = |N|, z = |Z|$ and $z_{max} = |Z_{max}|$ for a given (γ, β)
          $C_p = \langle Gain, B_l, B_r, N_l, N_r, N_c, Z, Z_{max} \rangle$
          $\lambda \leftarrow \lambda \bigcup \{C_p\}$
        end if
      end while
      Increment $level\_no$
      Enqueue($\varnothing$)
    end if
  end while
  Increment r
end while
Return optimal ms-cut $C_{max} \in \lambda$ with maximum *Gain*

trials. This bipartitioning method uses the same recursive framework presented in Algorithm 6 in order to identify a set of optimal monotone staircases recursively for the entire floorplan in $O((n^2 + nk) \log n)$ time.

5.5 Experimental Results

In order to verify the correctness and efficiency, we tested our algorithms presented in this chapter on MCNC and GSRC floorplanning benchmark circuits summarized in Table 5.1. The floorplan instances for each circuit were generated using *Parquet* tool [14] with random seeds. These algorithms were implemented in C programming language and the experiments were run on a Linux platform (2.8GHz, 4GB RAM).

Table 5.1: MCNC and GSRC Floorplanning Benchmark Circuits [14]

Suite	Circuit	#Blocks	#Nets	Avg. NetDeg
MCNC	apte	9	44	3.500
	hp	11	44	3.545
	xerox	10	183	2.508
	ami33	33	84	4.154
	ami49	49	377	2.337
GSRC	n10	10	54	2.129
	n30	30	147	2.102
	n50	50	320	2.112
	n100	100	576	2.135
	n200	200	1274	2.138
	n300	300	1632	2.161

In this experimental setup, we used a range of values for the trade-off parameters $\gamma \in [0.1\ 0.7]$ and $\beta \in [0.0\ 0.3]$, varying both (γ, β) in steps of 0.1 such that $\gamma + \beta <= 1$ is satisfied. In order to observe the performance of the proposed bipartitioners, we considered four different floorplan instances for each benchmark circuit. As stated earlier, these floorplan instances were generated by Parquet Floorplanning tool [14, 113] with random seed. We further state that, the experiments were conducted for area balanced optimization only. This is due to the fact that the areas of the blocks in each circuit are not uniform and hence not suitable for number balanced optimization. This fact has already been noticed in the results in terms of the respective *Gain* values for area and balanced bipartitioning methods in Section 3.5 of Chapter 3.

5.5.1 Bipartitioning Results

First, we ran the proposed BFS based floorplan bipartitioning method for bend minimization GenMSCut-Bend (refer to Algorithm 5) for specified (γ, β) pairs on all four floorplan instances of a circuit. In the plots, we tagged the results of this method as *BFS* mode. The same experiments were run on a modified version of the DFS based bipartitioner GenMSCut_DFS (see Algorithm 3) presented in Chapter 3, results of this method being highlighted as *DFS* mode, by suitably modifying it for bend minimization. This modification incorporates Equation 5.1 for evaluating *Gain* values for a given bipartition as well as all three objectives defined in Section 5.3, as compared to Equation 3.1 used in Chapter 3. In addition to these two greedy methods (BFS and DFS modes), the same set of experiments were also conducted for the proposed randomized floorplan bipartitioner GenMSCut-Rand presented in Algorithm 7 and tagged as *RAND* mode. The results from these three variants of the floorplan bipartitioners yielding minimal bend monotone staircases are compared with the BFS based greedy bipartitioner in Section 3.4, namely GenMSCut (see Algorithm 1), which do not consider bend minimization as an objective. We call this method as *BFS-NB* mode.

5.5.1.1 Comparing Greedy Methods: with and without Bend Minimization

We first study the results from BFS based greedy bipartitioning methods for both with and without bend minimization, namely BFS and BFS-NB, in Table 5.2. In this study, we compare the values of the area balance ratio, netcut ratio and the maximum *Gain* for each of the benchmark circuits along with the CPU time required by each of these two methods for the bipartitioning of each floorplan instance of a circuit recursively. In this experiment, we use $\gamma = 0.4$ for BFS-NB mode and aforementioned (γ, β) pairs for each of the circuits in BFS mode. For BFS mode, we take an average of the individual parameter values over all (γ, β) pairs and present them in the respective columns. However, we do not present the bend count values for BFS-NB mode in this study and restrict to only area balance ratio, netcut ratio and the maximum *Gain* values for both the modes. In the subsequent section, we present a detailed study of bend count for the other two modes along with these modes. The results in this table show that, BFS mode has marginally better net cut ratio than BFS-NB mode. However, it yields poor area balance and maximum

Gain than the other. The runtime values for the respective BFS-NB and BFS modes obey their relevant theorems on their recursive frameworks (refer to Theorems 1 and 3 for BFS-NB and BFS modes respectively). The corresponding values show that BFS takes $3.6X$ time than BFS-NB due to the process of identifying the bends in the monotone staircases which lacks in BFS-NB. For each parameter values presented in this table, in terms of normalized geometric mean in the last row, we highlight the superior values, e.g., maximum area balance ratio in case of BFS-NB mode and netcut ratio for BFS mode.

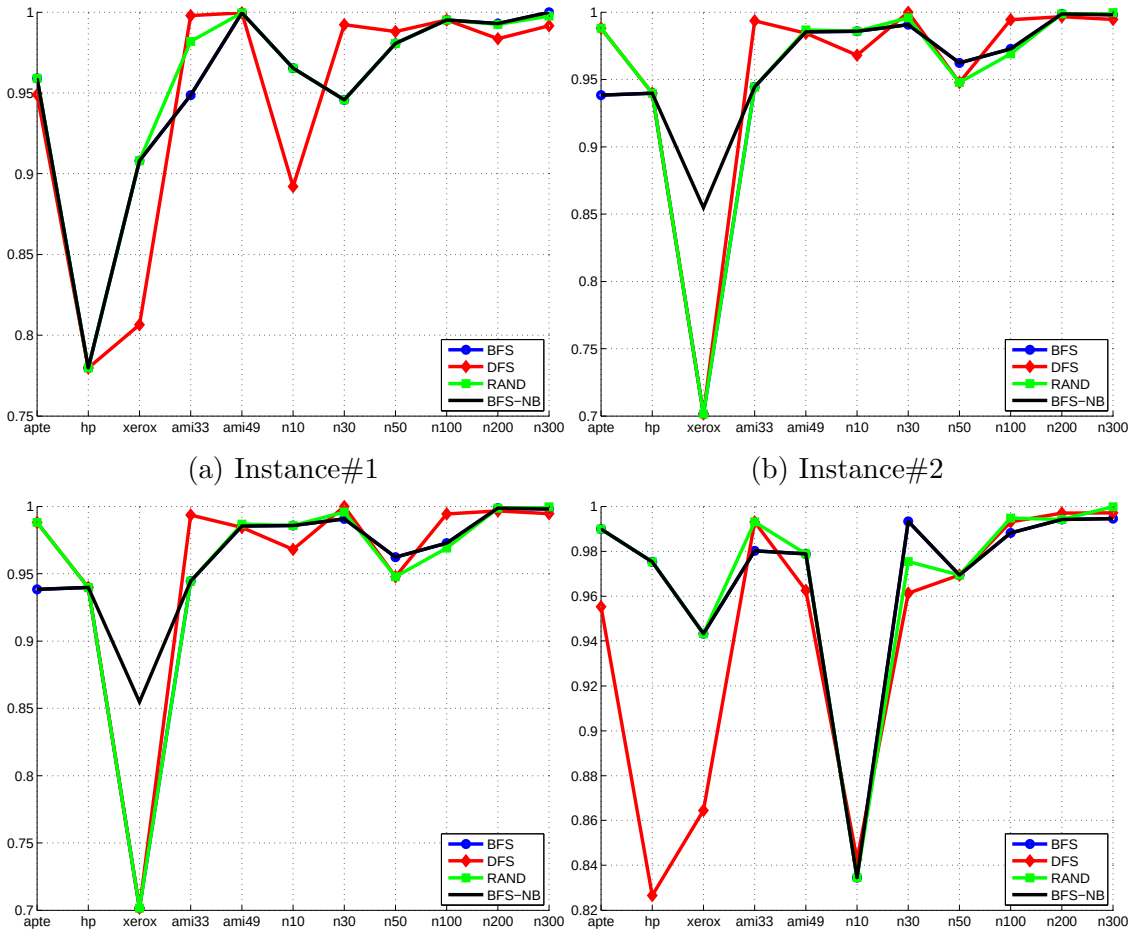
Table 5.2: Comparing Bipartitioning results between BFS vs BFS-NB

Circuit	Balance Ratio		Netcut Ratio		Max *Gain*		Runtime (sec)	
	BFS	BFS-NB	BFS	BFS-NB	BFS	BFS-NB	BFS	BFS-NB
apte	0.835	0.924	0.728	0.670	0.929	0.970	0.005	0.005
hp	0.741	0.908	0.607	0.784	0.886	0.963	0.006	0.003
xerox	0.826	0.903	0.588	0.568	0.924	0.961	0.011	0.009
ami33	0.959	0.961	0.593	0.580	0.984	0.984	0.033	0.014
ami49	0.931	0.990	0.497	0.527	0.972	0.996	0.107	0.023
n10	0.840	0.908	0.647	0.528	0.936	0.963	0.005	0.008
n30	0.948	0.979	0.515	0.558	0.979	0.992	0.031	0.006
n50	0.971	0.978	0.529	0.548	0.988	0.991	0.124	0.050
n100	0.988	0.988	0.541	0.530	0.995	0.995	0.803	0.062
n200	0.996	0.995	0.523	0.537	0.998	0.998	7.841	0.432
n300	0.996	0.997	0.528	0.553	0.998	0.999	21.945	0.656
Norm. Geo Mean	0.949	**1.000**	**0.988**	1.000	0.979	**1.000**	3.601	**1.000**

5.5.1.2 A Detailed Study on the Objectives

Now we present a detailed study for all the objectives and *Gain* function for: (i) BFS, (ii) DFS, (iii) RAND, and (iv) BFS-NB modes. The corresponding bipartitioning results are presented (a) in Figure 5.7 for measuring the quality of balance in area of each partition (L,R partitions), (b) in Figure 5.8 for normalized bend count, (c) in Figure 5.9 for normalized net cut, and (d) in Figure 5.10 for evaluating the maximum *Gain* value (see Equation 5.1) respectively for all four floorplan instances (namely Instance#1, #2, #3, and #4) of a given benchmark circuit.

Area Balance Ratio: The results on area balance between either partitions are presented in Figure 5.7 (a) - (d) for the respective floorplan instances. From these plots, we notice that depending on the floorplan instance and the number of blocks and the area distribution among the blocks, different modes dominates over the other

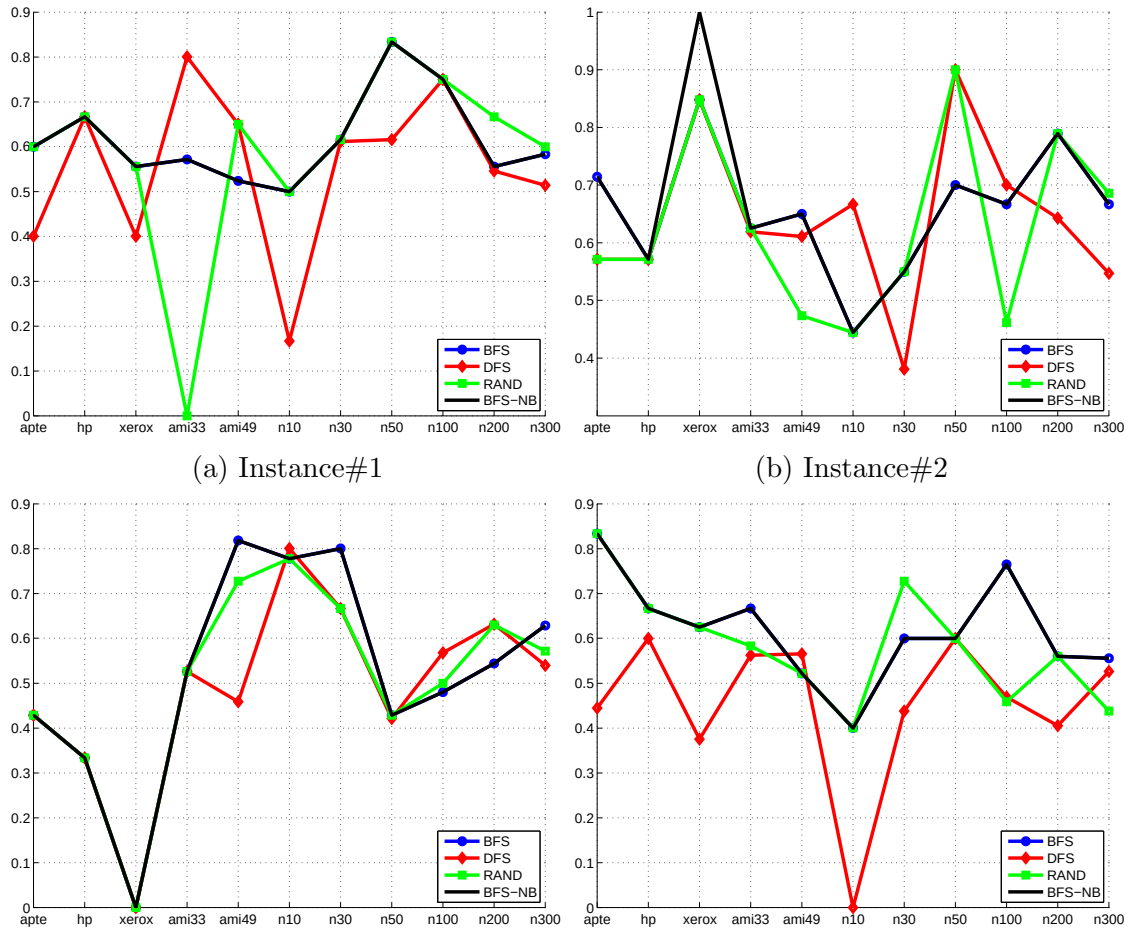


(a) Instance#1 (b) Instance#2

(c) Instance#3 (d) Instance#4

Figure 5.7: Average of Area Balance Ratio (*balr*) taken over all (γ,β) pairs for different floorplan instances

in different cases. For the circuits with larger block count, BFS, RAND and BFS-NB show better area balance ratio against DFS mode. Among these three modes, RAND mode is seen to dominate in case of most of the floorplan instances. This is due to the fact that RAND method can explore more solutions during the specified number of trials guided by the proposed neighbor search technique. With larger block count, this method ensures its higher randomization in the neighborhood indexing for picking an adjacent vertex. This potentially helps in exploring a optimal monotone staircase wave-front with higher area balance ratio, as per the objective declaration. With even higher number of blocks in a floorplan, RAND mode has the potential to cater an even better solution with higher area balance, ideally up to 1 for perfect balance.



(a) Instance#1 (b) Instance#2

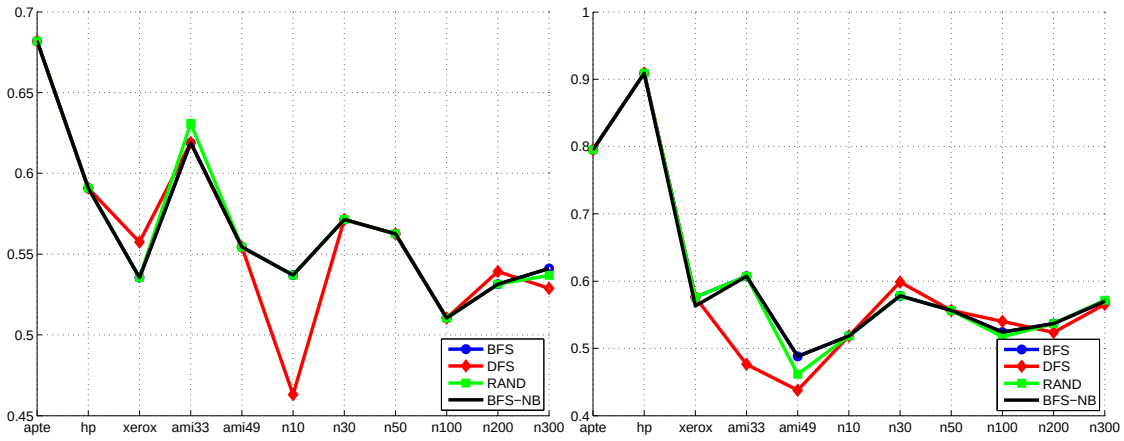
(c) Instance#3 (d) Instance#4

Figure 5.8: Average of Bend Ratio (z/z_{max}) taken over all (γ,β) pairs for different floorplan instances

Bend Ratio: The objective declaration in Section 5.3 demands that the number of bends in a monotone staircase is to be minimal. Our results on different floorplan instances of a given circuit show that a particular instance dominates the others. We also notice that for some floorplan instances, the bend ratio is 0. This implies that the partition fails to obtain a minimal bend staircase while satisfying other objectives. On the other hand, a value of 1 for the bend ratio implies maximum number of bends in a monotone staircase, i.e., when z becomes equal to z_{max}. But, these instances of 0 or 1 values for normalized bend count are obtained in a specific mode only for a few circuits with very small block count and a particular floorplan instance of it. For example, Instance#1 of *ami*33 in RAND mode only, Instance#3 of *xerox* in all modes including BFS-NB and Instance#4 of $n10$ in DFS modes show these kind of

behaviors.

Looking at the results, we make another important observation that RAND mode is able to identify minimal bend count in certain floorplan instances of the circuits. This implies that the randomized bipartitioning method has the potential to identify a monotone staircase with minimal number of bends, given that we carry out the required number of trials for each circuit while considering bend minimization only. On the other hand, both BFS and DFS methods selectively yield lower bend count in some of the smaller circuits. Despite the fact that BFS-NB does not consider any bend minimization, but its bipartitions show results similar to that of BFS.



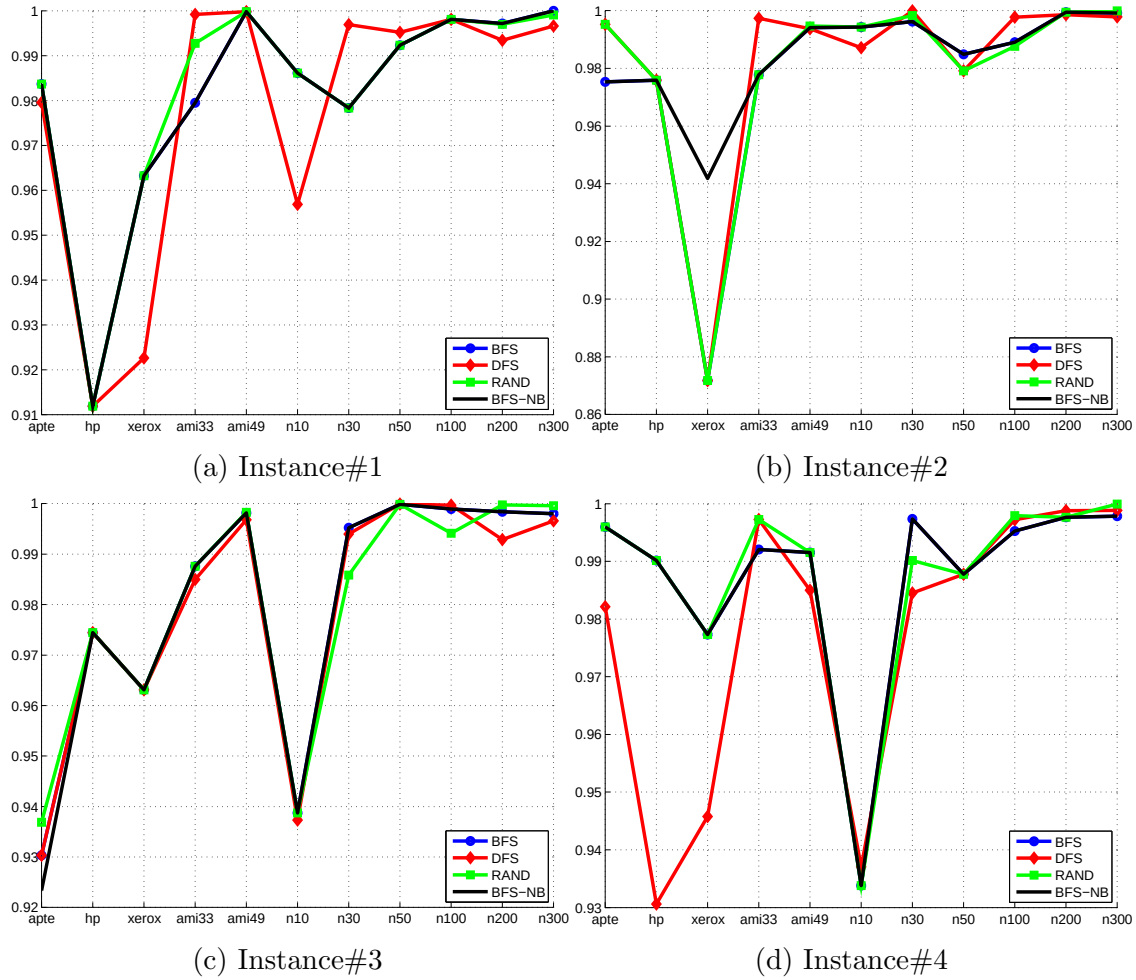
(a) Instance#1 (b) Instance#2

(c) Instance#3 (d) Instance#4

Figure 5.9: Average of NetCut Ratio (k_c/k) taken over all (γ,β pairs for different floorplan instances

Net Cut Ratio: As per our claim stated earlier in this chapter that our focus on net cut for a given bipartition is different than earlier maxflow based methods

[8, 102]. Unlike them, our graph model do not incorporate the net topology and the bipartitioning methods like BFS and BFS-NB run much faster. This is the reason that our model handles the partitioning of a net of any number of terminals (2 or more) equally well as compared to their model. The results on all the floorplan instances show that, the net cut information entirely depends on a given circuit and its corresponding floorplan topology (instance). As per the results obtained, BFS and BFS-NB methods are the most consistent in obtaining a minimal net cut value among the others; DFS and RAND dominate them in some of the cases.



(a) Instance#1 (b) Instance#2

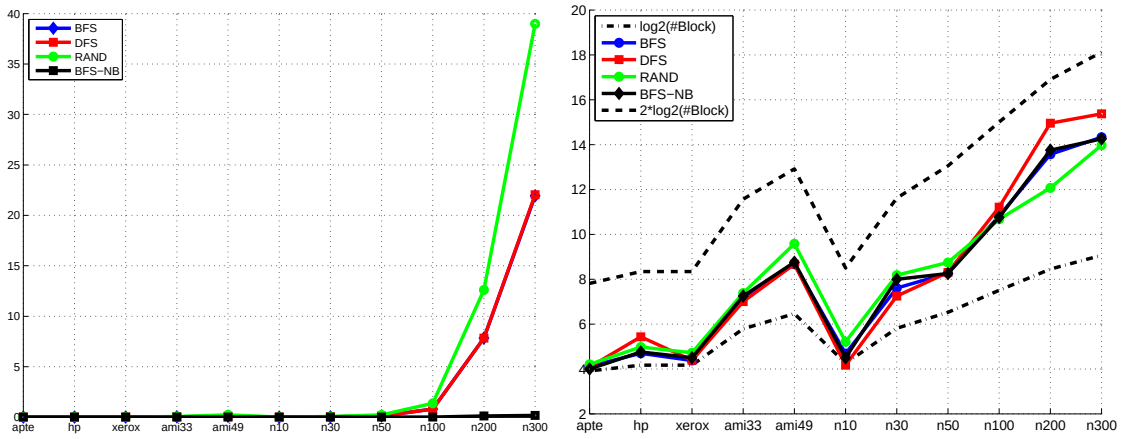
(c) Instance#3 (d) Instance#4

Figure 5.10: Average of Max *Gain* taken over all (γ,β) pairs for different floorplan instances

Maximum *Gain*: In Chapter 3, we presented the maximum *Gain* values obtained by BFS-NB and DFS-NB methods without considering bend minimization. In those results, we have seen that BFS mode was consistently better than DFS mode by a

small margin. In this experiment, we also notice a similar pattern in the variation of max *Gain* value for the different instances of the benchmark circuits. DFS mode gives better *Gain* values for a very few instances with smaller block counts whereas the values of *Gain* obtained by BFS and BFS-NB methods are consistently higher. Notably, the RAND mode gives better *Gain* values for most of the floorplan instances of the circuits with higher block count. This implies, that RAND mode has the potential to obtain an optimal monotone staircase for a given (γ,β), with a better scope of randomized neighbor search. This leads to a better optimal solution than BFS or DFS modes in terms of *Gain* values.

Runtime: The runtime results for BFS, DFS and RAND modes of the proposed floorplan bipartitioners are presented in Figure 5.11 (a), along with that for BFS-NB mode. As stated in Section 5.4, RAND mode is merely a constant times higher than the other two modes (BFS and DFS) and is prominent with larger circuits such as $n100$, $n200$ and $n300$.



(a) (b)

Figure 5.11: Plots for: (a) Average Runtime (sec), and (b) Average height of the MSC tree

Height of the bipartition tree: Earlier in this section, we have shown that the height of the MSC tree obtained by both BFS and DFS based algorithms without considering bend minimization is $O(\log n)$, where n is the number of blocks. This implies that our bipartition algorithms yield (nearly) balanced binary tree, which is also supported by the results presented in Figure 5.11 (b) for each circuit in case of bend minimization. The results show that the average height of the MSC tree for each circuit, taken over all four instances and for all (γ,β) pairs used in this experiment,

is contained within the tight bounds of $\log n$ and $2 \log n$.

5.5.2 Results on Early Global Routing

As stated in the earlier chapter, despite all the bipartitioning results presented so far, optimality of a set of monotone staircases for the entire floorplan is not truly reflected by the corresponding maximum *Gain* values (see Equations 3.1 and 5.1) and the parameters for individual objectives, specially bend minimization stated in this chapter. It requires an assessment of the optimization of those objectives for a given (γ, β) on via count in a given floorplan. We also study the impact of these bipartitioning results on other routing metrics such as routability, netlength and congestion, which is measured by a parameter called $wACE4$ using the edge based congestion (demand/capacity) parameters $ACE(x)$ prescribed in [15]. This evaluation is done by the proposed early global routing method STAIRoute presented in Chapter 4. We use the corresponding monotone staircases as the routing regions in the routing model of STAIRoute and route the nets through these regions in a number of specified metal layers. In this experimental setup, STAIRoute uses up to 8 metal layers using reserved layer model for layer assignment of the net segments. It is to be noted that we can not verify these bipartitioning results with any other early global routers as no such router exists as per our knowledge. We can not even use these results in the existing post-placement global routers as they fall in a different scope of operation in the existing PD flow and use a completely different routing paradigm. In Tables 5.3, 5.4 and 5.5, we present the routing results for netlength, via count and congestion respectively for the corresponding bipartitioning results on a given floorplan instance of the benchmark circuits. We observe that all these results correspond to 100% routability for all the benchmark circuits in all the three modes.

Results on Netlength: The results on netlength in Table 5.3 show that BFS and DFS methods with bend minimization show similar results as presented in Section 4.5. The netlength obtained for RAND mode is slightly higher than these two. However, without bend minimization, as in BFS-NB mode, the results are slightly higher than all the other three modes. These observations are also evident from the geometric mean values of the normalized netlength at the last row of the table. It shows that DFS method has the lowest value of 1.201, while BFS, RAND and BFS-NB have 1.207, 1.208 and 1.287 respectively. Alike the previous chapter, this normalization is done by the corresponding Steiner length of the nets for each circuit computed by FLUTE

[10]. Notably, Steiner length computation was done considering the coordinates of the pins in the net only and no routing blockage in the floorplan layout was considered.

Table 5.3: Average Netlength (in 10^3 μm) for different bipartitioning modes

Circuits	BFS	DFS	RAND	BFS-NB	Steiner Length [10]
apte	402.20	400.40	400.56	398.14	338.63
hp	201.98	202.81	207.79	202.00	123.72
xerox	346.91	337.83	348.54	716.92	304.38
ami33	110.92	110.78	110.31	111.13	92.33
ami49	1861.60	1817.69	1804.56	1809.09	1629.26
n10	19.84	19.84	19.84	19.84	16.63
n30	59.26	58.72	60.41	59.59	49.37
n50	151.59	155.38	151.63	151.60	125.02
n100	250.60	249.30	251.24	251.46	212.11
n200	429.29	425.87	429.37	429.85	381.02
n300	791.59	787.47	783.51	792.14	699.01
Norm. Geo. Mean	1.207	**1.201**	1.208	1.287	1.000

Results on Via Count: In Table 5.4, we present the via count values for all the modes along with the normalized geometric mean values for each mode. In this case, the normalization is done with respect to BFS-NB mode. These results also show that BFS and DFS yield similar via count, with 0.910 and 0.911 normalized geometric mean among all the circuits, while RAND mode has slightly higher values (0.923). These results also show that these three bend minimization methods yield 7-9% fewer vias than BFS-NB.

Later in this section, we present a detailed study on via count for some of the biggest circuits such as $n100$, $n200$ and $n300$ for different (γ,β) pairs in all three methods of bend minimization, i.e., BFS, DFS and RAND. In this study, we omit BFS-NB mode as they are shown to yield inferior via counts as per the results obtained in Table 5.4.

Results on Worst Average Congestion: Another important early global routing metric we consider in our experiment is the worst congestion in the edges of our routing graph in any routing layer (total 8 in this experiment). This metric is measured by a new parameter $wACE4$ computed using the parameters presented in [15] in order to showcase the effectiveness of the bipartitioning results obtained in each of the modes. From the normalized geometric mean values given in the last row of

Table 5.4: Via Count for different bipartitioning modes

Circuits	BFS	DFS	RAND	BFS-NB
apte	404	404	404	404
hp	502	507	539	502
xerox	404	404	416	1190
ami33	1156	1155	1157	1156
ami49	3598	3442	3507	3423
n10	176	176	185	176
n30	929	944	961	937
n50	3196	3346	3216	3194
n100	6776	6576	6738	6748
n200	17950	18276	17919	18016
n300	29607	29439	29602	29639
Norm. Geo. Mean	**0.910**	0.911	0.923	1.000

this table, we see that RAND mode yields 9% less congestion as compared to the congestion values obtained in BFS, DFS and BFS-NB modes. These values were normalized with respect to BFS-NB. It also shows that BFS and DFS modes has negligible improvement over BFS-NB as dictated the mean values. Despite that, all the modes conform to the fact that the congestion model in the proposed early global router STAIRoute presented in CHapter 4 does not allow congestion in any routing region to go beyond 100% mark in any of the routing layers.

Table 5.5: Worst Average Congestion ($wACE4$ [15]) for different bipartitioning modes

Circuits	BFS	DFS	RAND	BFS-NB
apte	0.894	0.894	0.894	0.901
hp	0.991	0.708	0.248	0.991
xerox	0.834	0.852	0.621	0.776
ami33	0.967	0.966	0.966	0.967
ami49	0.635	0.749	0.743	0.679
n10	0.644	0.644	0.935	0.644
n30	0.653	0.673	0.548	0.659
n50	0.838	0.909	0.907	0.818
n100	0.988	0.980	0.983	0.990
n200	0.481	0.489	0.469	0.489
n300	0.639	0.653	0.815	0.656
Norm. Geo. Mean	0.997	0.996	**0.908**	1.000

5.5.2.1 A Detailed Study on Via Count

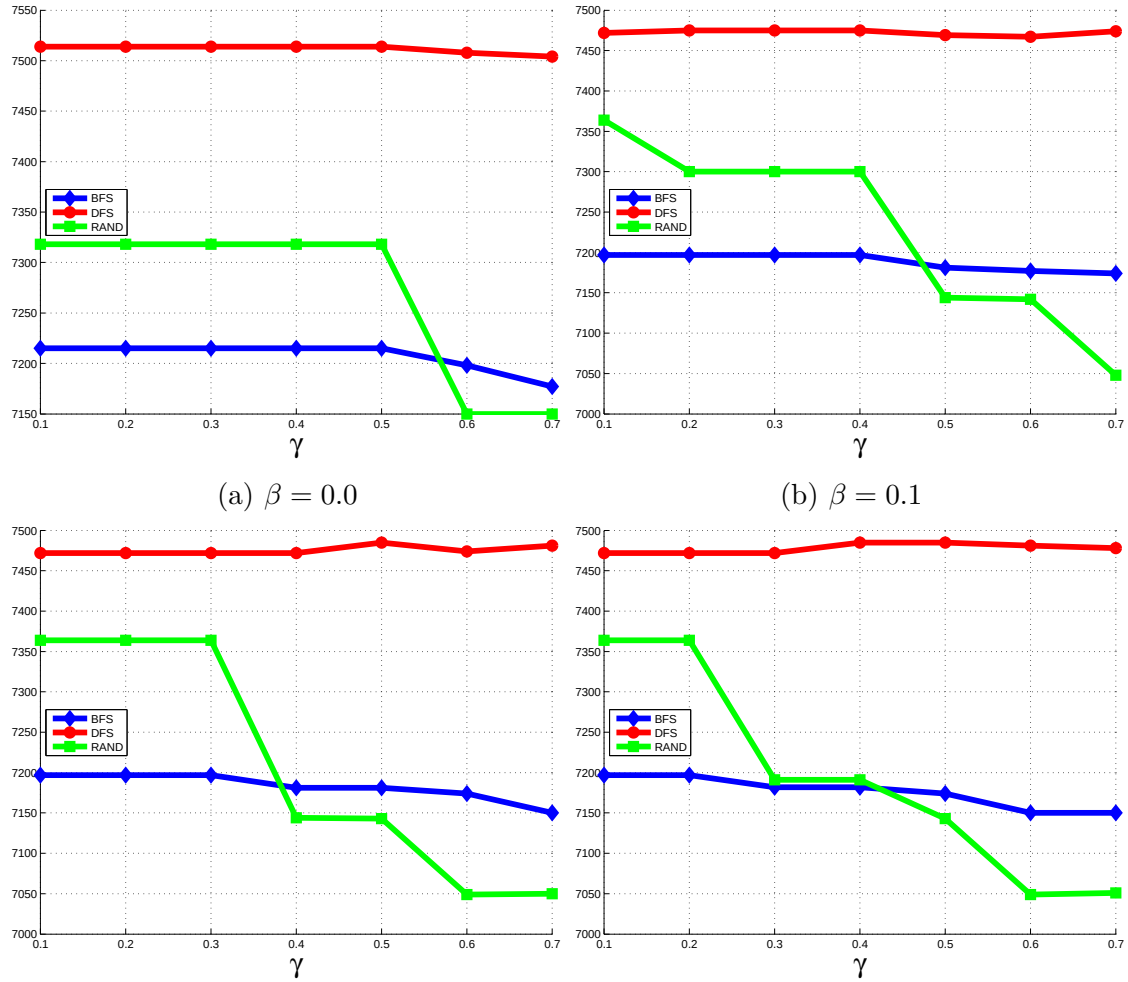
In this section, we present a detailed study on the variation of via count for a given floorplan instance of some of the benchmark circuits versus a set of (γ, β) values used in this experimental setup. Figure 5.12 shows four different scenario of via count variation among BFS, DFS and RAND modes with respect to different γ values and a specific β value in each. From the first plot in this figure for $\beta = 0.0$, we see that DFS yields higher via count than BFS and RAND, and remains almost constant for all γ values. On the other hand, BFS dominates over RAND with fewer via count for γ values up to 0.5, while RAND succeeds over it for higher γ. With the increase in β, this pattern recurs with RAND dominating with higher γ values for nonzero β values. In all the instances of β, DFS remained to have the worst via count.

Similarly, the via count plots for $n200$ in Fig. 5.13 show that BFS mode continues to yield the best via count as compared to DFS and RAND modes for all pairs of (γ, β) except for a few when β is greater than 0.1 and a higher γ value such as 0.7. In case of those few instances, DFS mode dominates in the lowest via count. The via count variation in BFS mode shows a non-decreasing pattern with the increase in γ for any β value, more prominent for $\beta > 0.1$, while DFS mode shows negligible variation in via count for all pairs of (γ, β). In these results, we do not see the best results from RAND mode except for nonzero β values where it is better than DFS mode. For $\beta = 0.0$, RAND mode shows the worst via count than the other two modes.

In the last set of plots for via count in Fig. 5.14, we present the results for the largest circuit $n300$ for four β values in the respective plots. When $\beta = 0.0$, DFS mode shows the best results for all γ values, while RAND shows the worst. With the increasing β values, RAND mode start to dominate over the other two modes for up to some specific γ values; 0.3, 0.5 and 0.4 when the respective β values are 0.1, 0.2 and 0.3. Beyond these γ values, in each case of nonzero β, DFS shows the best via counts.

5.6 Chapter Summary

In this chapter, we present an improved recursive floorplan bipartitioning framework for defining a set of minimal bend monotone staircase routing regions in a given floorplan, intended to address an early version of unconstrained via minimization (UVM) problem using the early global routing framework presented in Chapter 4.



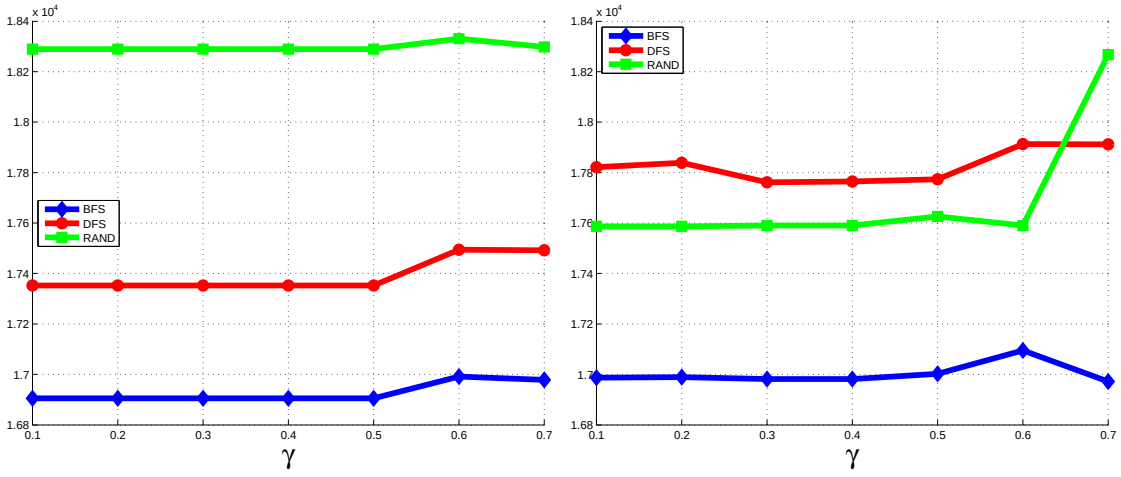
(a) $\beta = 0.0$ (b) $\beta = 0.1$

(c) $\beta = 0.2$ (d) $\beta = 0.3$

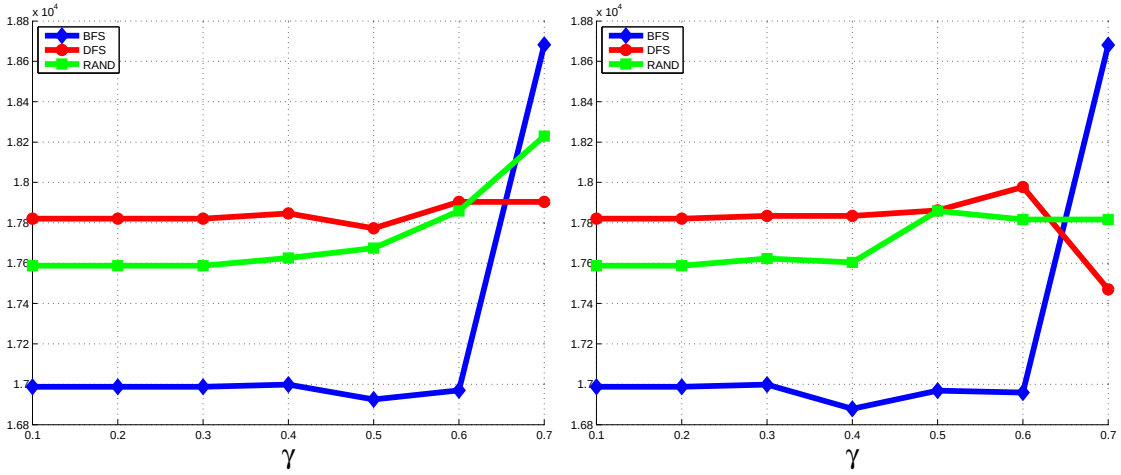
Figure 5.12: Via count for a floorplan instance of $n100$ circuit vs. (γ, β)

This method incorporates a new objective of bend minimization in this multi-objective floorplan bipartitioning problem. First, we present BFS/DFS based greedy methods followed by a randomized neighbor search based staircase wave-front propagation approach.

Unlike Chapter 3, the proposed BFS/DFS based bipartitioning approaches yield a sequence of $n-1$ monotone staircases for a (sub)floorplan of n blocks at that level, at any level of the bipartition hierarchy (MSC tree). An optimal monotone staircase (solution) is chosen as the one with maximum *Gain* value from that sequence. Notably, the number of possible monotone staircases in any floorplan is exponentially large. In order to increase the solution space of $n-1$ obtained by these greedy approaches, we



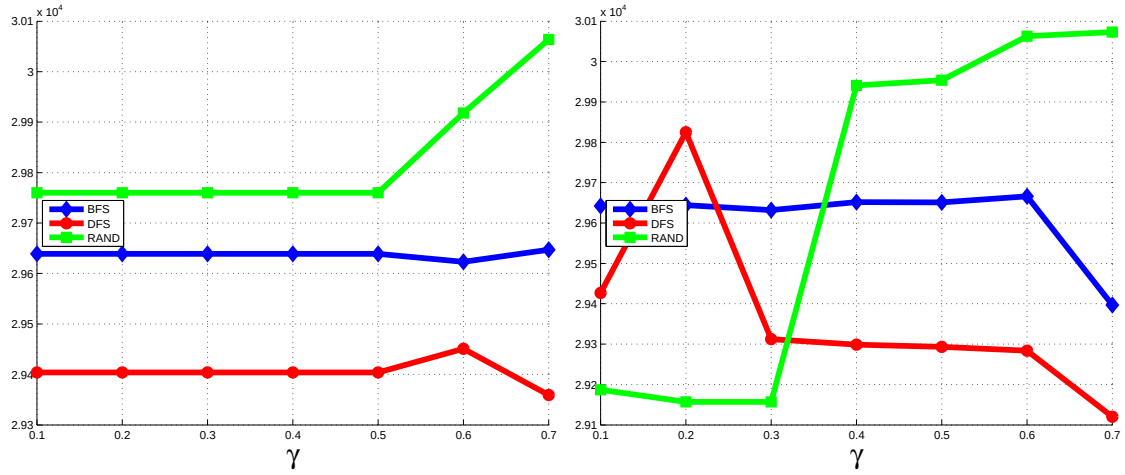
(a) $\beta = 0.0$ (b) $\beta = 0.1$



(c) $\beta = 0.2$ (d) $\beta = 0.3$

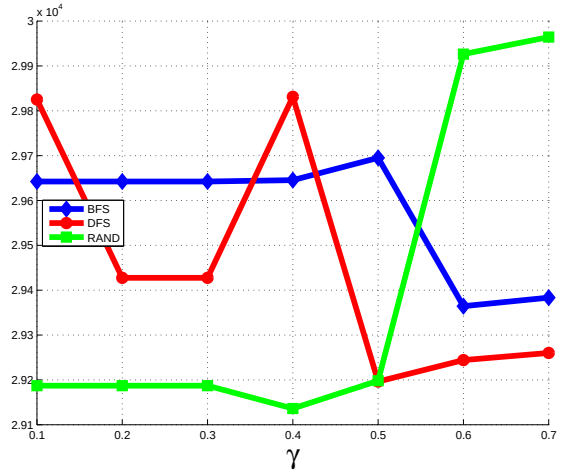
Figure 5.13: Via count for a floorplan instance of $n200$ circuit vs. (γ, β)

present a new bipartitioning method that randomly explores the neighborhood of a vertex in the BAG and generates a sequence staircase of wavefronts in the floorplan, obeying the monotone staircase property defined in Lemma 1 [102]. In this method, multiple trials are attempted for generating a new sequence of monotone staircases during each trial. These sequences are not necessarily disjoint and thus can have one or more common solutions. Union of all such sequences creates a solution space substantially larger than those obtained by the greedy methods, although size of this space depends on the number of trials used. An optimal solution is chosen from this enlarged solution space as the best one with maximum *Gain* for a specified (γ, β) trade-off pair.

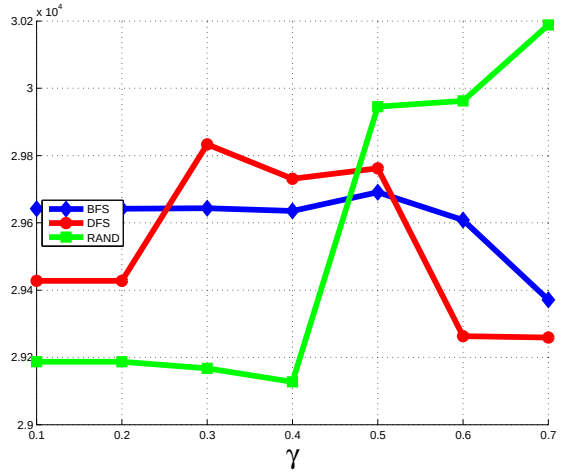


(a) $\beta = 0.0$

(b) $\beta = 0.1$



(c) $\beta = 0.2$



(d) $\beta = 0.3$

Figure 5.14: Via count for a floorplan instance of $n300$ circuit vs. (γ, β)

Experimental results show that RAND mode is potentially capable of obtaining better bipartitioning results on any floorplan based on a specific trade-off among the objectives, emphasizing on identifying minimal bend monotone staircase routing regions with non-zero β values. These bipartitioning results used in the proposed early global router STAIRoute presented in Chapter 4. The corresponding routing results show significant reduction in wirelength, via count and even in congestion, as compared to the bipartitioning approaches in Chapter 3 without considering the objective of bend minimization in the corresponding optimization problem. Therefore, this framework along with STAIRoute has the potential to obtain minimal via early global routing solutions on any floorplan.

Chapter 6

DFM aware Early Routability Assessment

6.1 Introduction

In modern IC fabrication process nodes such as those below $65nm$, severe yield loss has been observed that are primarily caused by design for manufacturability (DFM) issues. One such issue is chemical mechanical polishing (CMP) which induces surface irregularities during the copper metalization process. Due to different hardness factors of copper and dielectric materials, over-polishing and under-polishing of these materials occur across the metal layers [90] causing structural faults like open and shorts due to lack of sharpness around depth of focus (DOF) of the lithographic illumination system. Electrical characteristics of the metal wires such as resistance and capacitance also become unpredictable impacting both power and timing. So far, dummy metal fills insertion [91, 92] has been a popular method to alleviate these issues as these tend to (i) reduce the unpredictability in the electrical properties of the metal interconnects, and (ii) improve uniformity in metal density across the layers causing fewer CMP variations. But, these dummy fills pose serious burden on the power/ground network due to induced cross-talk from high coupling capacitance across the layers and also increased IR drops [93]. Therefore, a uniform feature density distribution including an effective placement of cells and pins [4, 94], and a suitable global/detailed routing solution with uniform wire distribution across the routing layers [52, 88] appears to be more practical than mere dummy fills.

In order to reduce CMP irregularities, recent works in [52, 88] emphasized on con-

trolling metal (wire) density during the routing phase by estimating the pin density in a routing region. While [88] proposed wire density driven global routing, [97] proposed a maze routing emphasizing wire distribution. A similar work in [52] presented a two pass top-down routing framework implementing wire density driven global and detailed routing in each pass. In these works L/Z shaped pattern routing was used in order to minimize via count. The authors in [88] also demonstrated a relation between congestion and wire density in a routing region considering variable metal width in a given layer due to both power and signal wires. Moreover, while [88] emphasized on wire density control within a routing tile, [52] focused on inter-tile wire density gradient to address large scale variation. But, none of them considered both intra-bin (local) and inter-tile (global) wire density variation together. In another work in [4], the authors proposed a predictive CMP model based probabilistic routing method in order to estimate the metal density and subsequently integrated it in their iterative placement tool for a CMP-aware routability driven placement solution. A top-down CMP aware placement method in [94] modeled wire density along the edges of the placement bins based on a well defined balance between hyper-edge cut cost and wire density cost.

6.2 Uniform Wire Distribution in a Floorplan

In this section, we present an early global routing framework that facilitates uniform wire distribution across the routing layers in a given floorplan. This framework, alike the early global routing framework STAIRoute, uses minimal bend pattern routing through the monotone staircase routing regions obtained for the given floorplan (see Chapter 5). In the proposed framework, after the placement is done respecting the floorplan topology, global routing of the nets is then to be performed for those nets connecting the hard and the soft blocks only, treating the soft blocks as *hard* as the macros. During soft block formation, the given standard cell netlist connecting the standard cells and the macros is modified, and those within a cluster (soft block) are masked. Typically, the macros have well defined pins at their boundaries. But there are no well defined boundary pins for these soft blocks, and are usually assumed to be at the center of the respective blocks. In recent day's design flow, a new concept called *virtual pin* evolved that represent a set of pins that render a kind of gateway to the nets that enter or exit these soft blocks. However, the nets contained entirely

within a soft block do not have any contribution towards any of the virtual pins at the boundaries of that soft block.

Unlike the early global routing method STAIRoute presented in Chapter 4, the salient features of this method are as follows: (a) a new congestion model for assigning a routing penalty on the edges of the routing graph (refer to Chapter 4) across the routing layers considering both terminating and non-terminating nets in a routing region, (b) a heuristic method to assign the pins at the boundary of the blocks in the given floorplan when they are assumed to be at the center of the blocks during floorplanning, and (c) a hierarchical net routing order based on the floorplan bipartition tree (see Chapter 3). The proposed congestion model guides the router to route the nets more uniformly both within the same routing layer as well as across different routing layers. More uniformity in wire distributions implies reduced inter-layer capacitance variation due to CMP process while reduced average congestion implies reduction coupling noise due to intra-layer capacitance.

6.2.1 The Proposed Method: WDGRoute

Before discussing the proposed uniform wire distribution driven early global routing method [118], we consider a small example in Figure 6.1 to showcase two different routing instances in a routing region. In this example, we identify two different categories of nets, namely (i) those nets terminating on at least one pin, and (ii) the nets those have no terminating pins in this region. The first category of nets are called *terminating nets*, while the others are referred to as *non-terminating* or *fly-by* nets. In this example, there are five nets that are bound to terminate on the six pins located in this region using some of the available tracks. Out of all the terminating nets, only one 2-terminal net has both the pins in this region. If this net were a 3 terminal net containing all the pins in this region having a total of 6 pins per say, then there would be three more nets that terminate in this region. This clearly indicates that the pin count is not a direct measure on the number of terminating nets.

In this example, the routing penalty for the first non-terminating net will be the same due to the cumulative effect of all the terminating nets, no matter whether the terminating nets have already been routed or not. Being local to this region, these static nets will always be routed before non-terminating nets, and hence be treated as static. Subsequent routing will be only for non-terminating nets abiding by a least cost routing path. We discuss the routing order of nets later in this section after

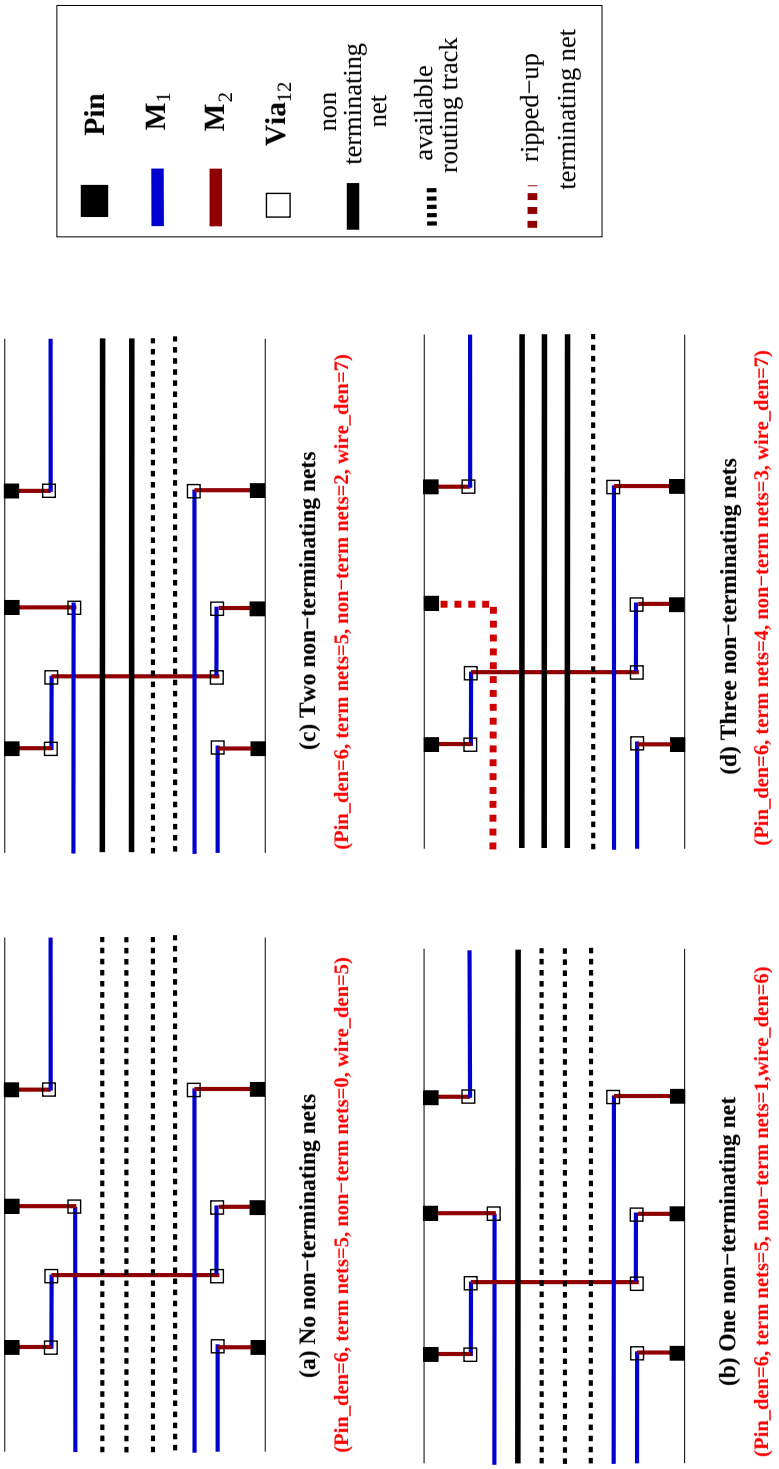


Figure 6.1: Different routing instances with 8 routing tracks in a routing region having (non-)terminating nets

elaborating on the rationale behind the new congestion model.

6.2.1.1 The Congestion Model

As discussed in Chapter 4, the congestion model in STAIRoute allows the routing demand in a given routing region up to its maximum routing demand, i.e, the routing capacity of that region in a given routing layer. The congestion results obtained for STAIRoute and this work on HB floorplanning benchmarks [16] (see the respective plots A and B in Figure 6.2(a)) show that the congestion distribution of STAIRoute is more skewed towards higher congestion values with significant standard deviation than the present work. This implies that the routing is less uniform across the layout. With more even distribution, our model intends to distribute the wires more uniformly across the routing layers considering the congestion scenario of terminating and non-terminating nets in the routing penalty (see plot B in Figure 6.2(a)).

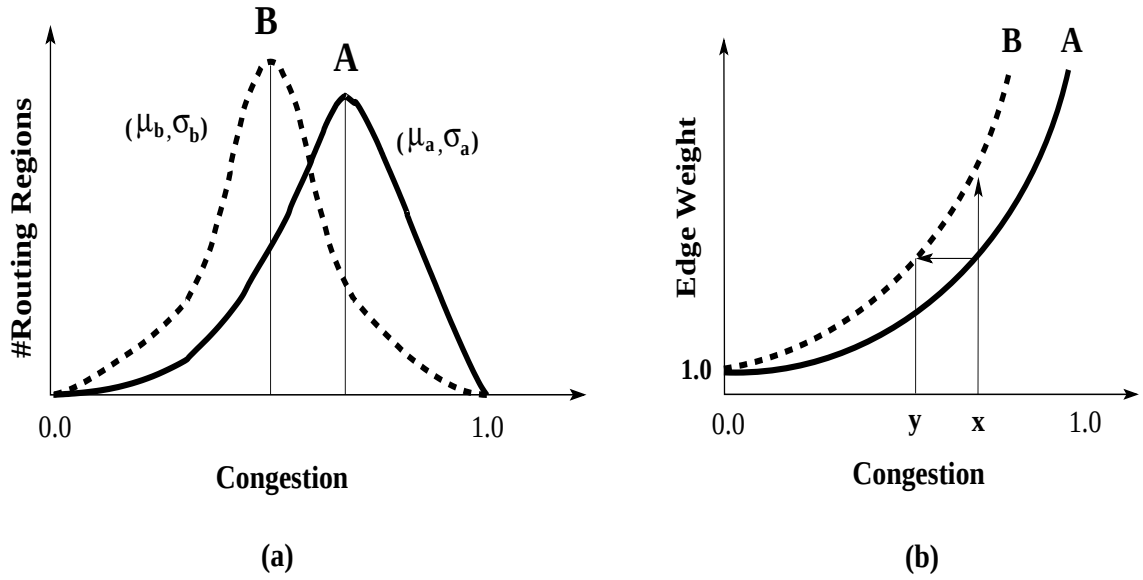


Figure 6.2: Comparison between STAIRoute (plot A), and this work (plot B) for: (a) congestion distribution, and (b) congestion penalty

In order to explain the proposed congestion model for uniform wire distribution, let us consider the two different instances in Figure 6.2 (b). The plot A represents the congestion model with only hyperbolic penalty without considering uniform wire distribution, while plot B represents the proposed congestion model. Considering no penalty due to uniform wire distribution, the congestion p_e in edge e has a value of $y(< 1.0)$ and is available for routing more nets. Subsequent routing through e will

increase p_e to a higher value, say x, where $y < x < 1.0$, with the corresponding routing penalty given by plot A. However, plot B will impart a huge penalty as compared to plot A for the same congestion value p_e in edge e. As per both the plots, the same routing penalty incurs much smaller congestion, i.e., y in plot B and x in plot A) in e. Therefore, our model restricts the net n_j to be routed through e, and encourages routing it through the next permissible metal layer or region say e'. This creates a congestion distribution similar to the plot B in Figure 6.2(b).

According to Equation 4.1, the routing cost for a net through a rectilinear segment e of a monotone staircase routing region is higher when the routing demand is higher, due to higher congestion penalty in the denominator (see plot A in Figure 6.2(b)). However, it is difficult to identify whether the penalty is contributed by the routing demand due to terminating nets or non-terminating nets or both. Therefore, we propose a new formula which considers both as below:

$$wt(e) = \frac{length(e)(1 + u_e^t/r_e^t)(1 + u_e^{nt}/r_e^{nt})}{(1 - u_e/r_e)} \tag{6.1}$$

where, $u_e^t(u_e^{nt})$ and $r_e^t(r_e^{nt})$ are the routing demands pertaining to terminating (non-terminating) nets and the capacity of a routing region e for a given metal layer M_i. Beside considering the overall routing demand as penalty in the denominator (similar to that in STAIRoute), Equation 6.1 also takes into account individual effect of terminating and non-terminating nets already routed in e in the factors in the numerator. For example, u_e^{nt} will be zero when no non-terminating nets routes through e in a routing layer. In higher routing layers, the routing penalty is relatively small during the initial iterations when very few (perhaps no) nets passing through them. Therefore, these are favorable for least cost routing, but with a possibility to have more via penalty. The proposed routing model amis to reduce both the worst average congestion [15] as well as the standard deviation, as depicted in Figure 6.2 (a) (plot B), by distributing the nets more uniformly across the layout in all metal layers.

6.2.1.2 Pin Distribution around a Block

In a floorplan, the pins of a block are not located at the boundary and are usually assumed to be located at the center. On the other hand, the pin density in the monotone staircase routing regions adjoining the block boundaries mandates the pins to be at the boundary of the blocks. Therefore, we propose a heuristic method

assigning the pins at the boundary using the net topology, as illustrated in Figure 6.3.

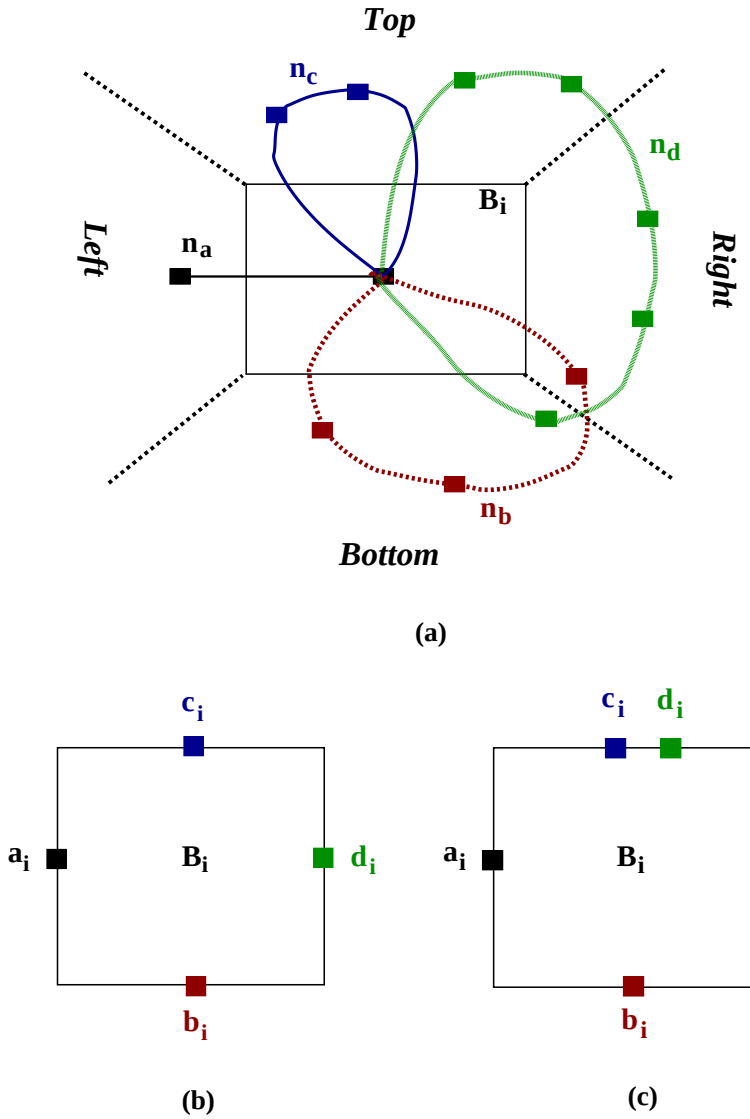


Figure 6.3: An example of pin assignment for a block B_i with the set of nets $\{n_a,n_b,n_c,n_d\}$: (a) relative positions of the pins on other blocks, (b) and (c) *two* possible outcomes of the proposed pin distribution method

In this method, the boundary of a block B_i is divided into four regions, namely *top, bottom, left* and *right*. By overlaying the topology of each net n_k connected to B_i on these regions, a ranking of these regions based on the number of pins is obtained. Subsequently, these regions are sorted a non-increasing order of the ranks. The boundary region assignment of a pin k_i of B_i connected to the net n_k is identified

as the region with the highest rank. If there is any tie on the highest rank value, a random tie-breaking mechanism is adopted. For example, for the net n_d in Figure 6.3, the ranking of top, right, bottom and left regions are 2, 2, 1 and 0 respectively. Therefore, the pin d_i of block B_i connected to n_d can either be placed on the top or the right boundary. Instances of pin assignment of B_i for a set of nets $\{n_a,n_b,n_c,n_d\}$ to it are also depicted in Figure 6.3.

6.2.1.3 Illustration for Uniform Wire Distribution

Now, we consider two routing instances of a net (t_a, t_h) in order to illustrate the proposed global routing method employing uniform wire distribution, in Figures 6.4 and 6.5. In the first instance (see Figure 6.4), the routing path comprising of net terminals and T-junctions is identified as $\{t_a,J5,J4,J9,t_h\}$ on the corresponding routing graph (GSRG) and seen to contained within the net bounding box of the terminals (t_a,t_h). Here t_a is treated as the source terminal of the shortest routing path while t_h pertains to the sink vertex. It is important to note that the pin junction edges $\{t_a,J5\}$ and $\{J9,t_h\}$ in the corresponding GSRG ensures an well defined pin accessibility. The routes for these edges use the same routing regions falling between the junction pairs $\{J1,J5\}$ and $\{J8,J9\}$ respectively, but use only a part of the entire horizontal/vertical region. In case of example (a) of this routing instance, the routing segments $\{t_a,J5\}$, $\{J5,J4\}$, $\{J4,J9\}$ and $\{J9,t_h\}$ use the corresponding horizontal/vertical wire segment routing using M_1, M_2, M_1, and M_2 metal layers respectively. Since the layer change occurs at the junctions only, e.g., $M_1 \rightarrow M_2$ at J_5, it requires three vias at J_5, J_4, and J_9 respectively with a minimum one layer change.

This example does not contain any wire distribution penalty during routing, similar to the early global router STAIRoute proposed in the previous section. On the other hand, due to uniform wire distribution cost in example (b), let us consider the routing penalty through $\{J5,J4\}$ at metal layer M_2 is very high (see Equation 6.1 and Figure 6.2) and hence the next permissible metal layer M_4 is used instead. As cited before, this reduces the congestion, measured as demand/capacity, in $\{J5,J4\}$ at M_2 at the cost of additional number of vias for routing the net segment through M_4. Since, this route for the net (t_a,t_h) is confined within its net bounding box and hence the wirelength does not exceed beyond its HPWL. In summary, we see that minimal length routing and minimal congestion scenario is obtained using the proposed congestion cost (penalty) due to uniform wire distribution, but with trade-off

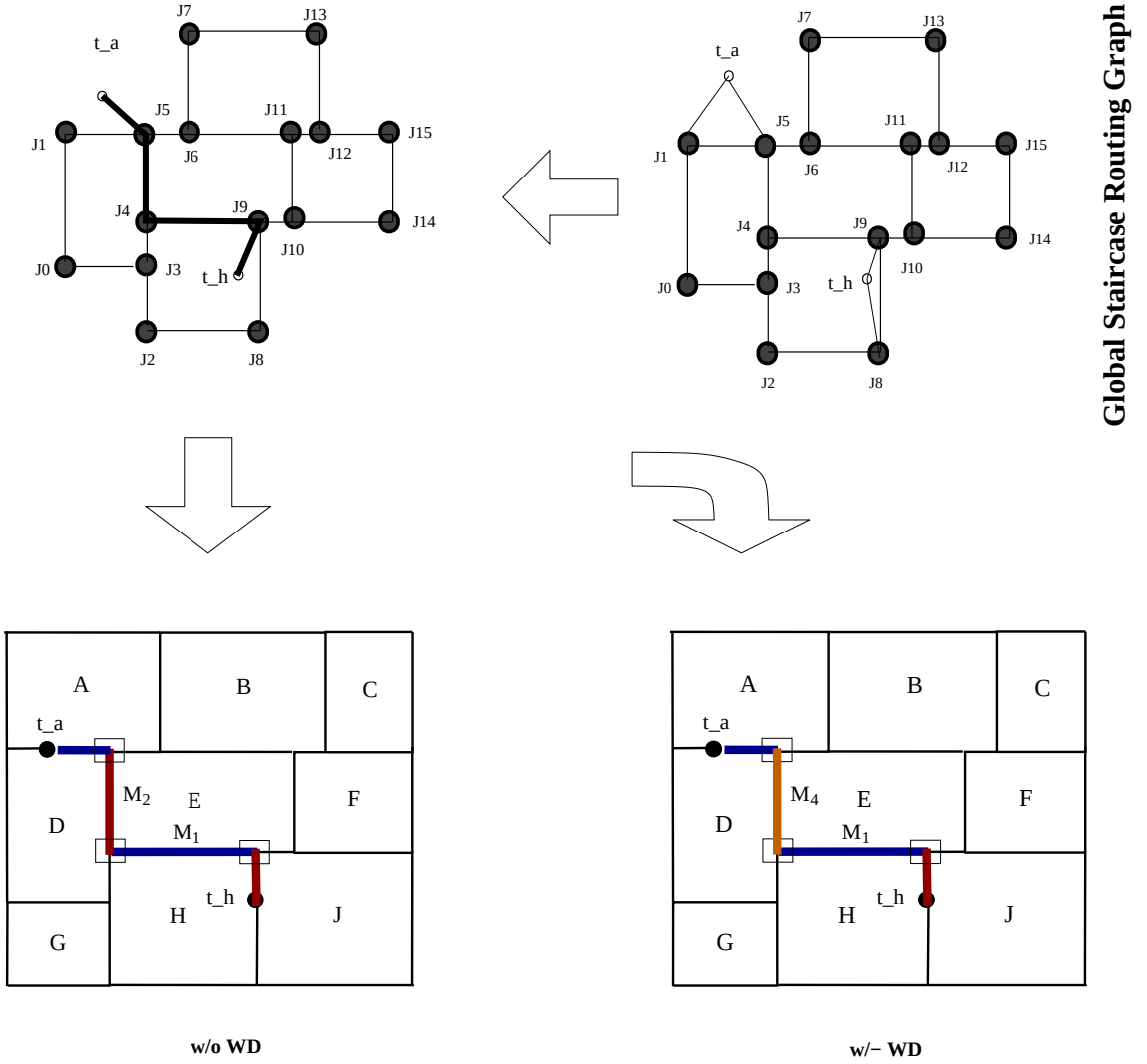


Figure 6.4: Multi-layer routing instance of a net (t_a, t_h) confined within the net bounding box: (a) without, and (b) with uniform wire distribution

of more number of vias.

In the second routing instance for the same net (t_a,t_h) (see Figure 6.5), we obtain a different routing path through the routing regions $\{t_a,J5\}$, $\{J5,J6\}$, $\{J6,J11\}$, $\{J11,J10\}$, $\{J10,J9\}$ and $\{J9,t_h\}$. Notably, this routing path is not contained within the net bounding box. Therefore, the routed netlength is more than HPWL of the bounding box. In this instance, starting with (t_a,$J5$) in M_1, this router subsequently chooses the segment $\{J5,J6\}$ instead of $\{J5,J4\}$ as the latter has exhausted all its routing capacity in all the permissible routing layers used by it. The segment $\{J5,J6\}$

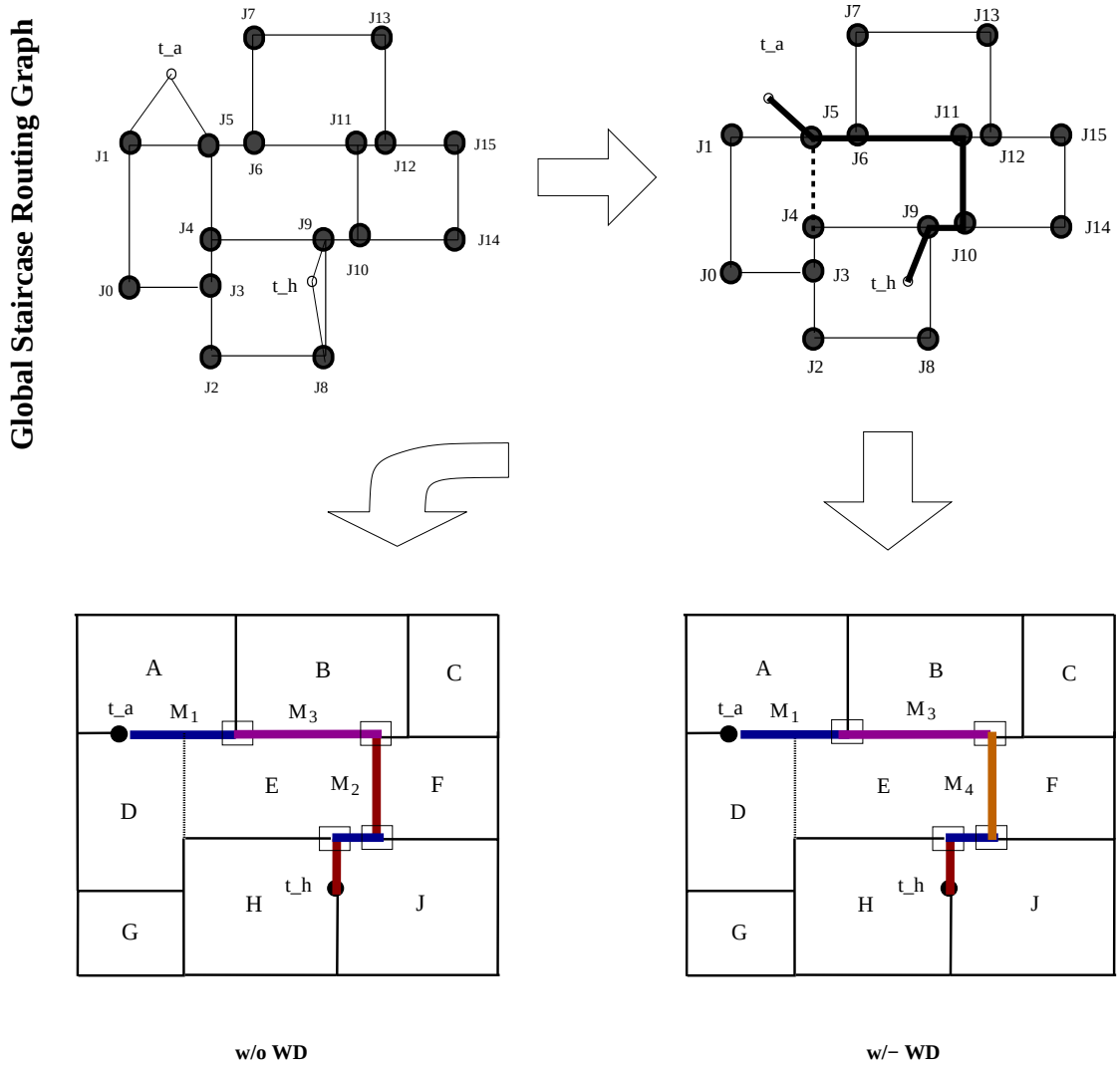


Figure 6.5: Multi-layer routing instance of a net (t_a, t_h) not confined within the net bounding box: (a) without, and (b) with uniform wire distribution

still has available space for routing through the layers M_1 and above, allowing more nets through it. Continuous effort by the router is to contain the routing path within the bounding box of the net, not allowing any detour. In this example, the router inherently yields detour, due to the resulting staircase topologies in the floorplan, by choosing the segment $\{J5,J6\}$ for routing in M_1 followed by $\{J6,J11\}$ in M_3 as the best candidates contained within the bounding box. These two segments give similar results due to adequate routing capacity and negligible different in routing cost due to wire density cost. In example (a) of this routing instance, the subsequent segment $\{J11,J10\}$ routes the net in M_2 when no wire density cost is considered.

Otherwise, as in example (b), this segment will push this net in the subsequent metal layer M_4 due to higher routing penalty considering wire distribution cost in M_2. This results in more via counts, but at no additional wirelength due to any further detour. Therefore, the congestion in $\{J11,J10\}$ at M_2 layer is reduced with wire density cost than when it is not considered. In summary, this routing instance is a corner case and occurs in many cases.

6.2.1.4 The Algorithm for Uniform Wire Distribution

In this section, we present the pseudo-code for the proposed early global routing method aimed at uniform wire distribution in a given floorplan, namely *WDGRoute*, in Algorithm 8. This framework is similar to STAIRoute presented in Chapter 4 by restricting the routing of the nets in the monotone staircase routing regions and employs Dijkstra's shortest path algorithm for identifying a monotonic routing path between a pair of terminals for a set of nets through them. These routing paths go through a number of metal layers using horizontal/vertical routing in alternate metal layers, using HV reserved layer model for routing. However, in STAIRoute, the pins were assumed to be at the center of the blocks and did not adopt the hierarchical routing order as proposed in this method. Moreover, the congestion model (see Equation 6.1) in WDGRoute includes a differentiated impact of the terminating and the non-terminating nets on the routing penalty as the routing progresses, while STAIRoute did not consider any such scenario. For multi-terminal net routing, a similar net decomposition method proposed in Section 4.3 is used to route each net segment between a pair of terminals (pins). This is followed by the Steiner tree identification method proposed in Chapter 4 in order to remove redundant net segments as well as reduce via count.

Notably, the global routes obtained by both STAIRoute and WDGRoute are strictly contained within the polygons shaped as monotone staircases across the floorplan. Therefore, all the nets pass through the regions bounded by the block boundaries, be it a soft block or a macro. Although these routing methods inherently constraint these routing paths within the net bounding box between a pair of terminals, there are few instances when a net has to route beyond its bounding box by taking the conjunctions of a set of monotone staircase routing regions. These routing instances usually occur when these early global routers search for a least cost routing path as per the respective congestion models and refrain to switch between multiple

Algorithm 8 WDGRoute: Uniform Wire distribution driven Early Global Routing

Inputs: A floorplan F with a set of blocks B, and nets N

Outputs: An early global routing on F with 100% routability, netlength, via count and uniform congestion (wire) distribution

/* Initialization/pre-processing Steps */

• Identify the monotone staircase routing regions by hierarchical floorplan bipartitioning e.g. GenMSCut-Bend and construct the junction graph G_j (See Chapters 4 and 5)

• Run the proposed pin distribution method for each block in the floorplan based on the netlist N

• Compute capacity of each monotone staircase routing region for both terminating and non-terminating nets applying Equation 6.1

• Obtain hierarchical routing order of the nets N based on the floorplan bipartitioning tree obtained for F

/* Routing Steps */

for all ordered nets $n_i \in N$ **do**
  Construct global staircase routing graph (GSRG) G_{ri} for n_i using G_j
  if n_i is a 2 terminal net **then**
    Identify the routing path for n_i using Dijkstra shortest path algorithm [109] on G_{ri} obeying Equation 6.1
  else
    Construct the node graph G_{ci} and identify its MST T_{ci}
    for all valid 2-terminal pairs $(t_j, t_k) \in T_{ci}$ **do**
      Identify the shortest routing path for the terminal pair (t_j, t_k) on G_{ri} obeying Equation 6.1
    end for
    Identify the *Staircase Steiner Point(s)*
  end if
  Compute netlength and via-count for n_i if the routing is successful
  Update routing demand (hence *congestion*) for all the routing regions along the routing path of n_i in the corresponding metal layers
end for
Compute the routing metrics and the congestion statistics for all the routing regions

layers for fewer vias along the path. In the subsequent section, we discuss a new routing model that attempts to mitigate these limiting cases of routing instances obtained by both STAIRoute and WDGRoute by allowing routing path over the blocks in specified routing layers.

6.3 Over-the-Block Early Global Routing

The existing physical design flow in Figure 1.9 depicts that global routing [17, 19, 20, 21, 22, 26, 27, 32] is performed after the placement stage on a set of nets connecting the macros and standard cells in a design. The first step in this framework is to partition the layout into m-by-m equal sized tiles called global routing bins, also known as *GCells*. These bins act as the vertices in the corresponding grid graph model as depicted in Figure 6.6 (a). Each edge in this graph corresponds to a pair of tiles that share a common boundary.

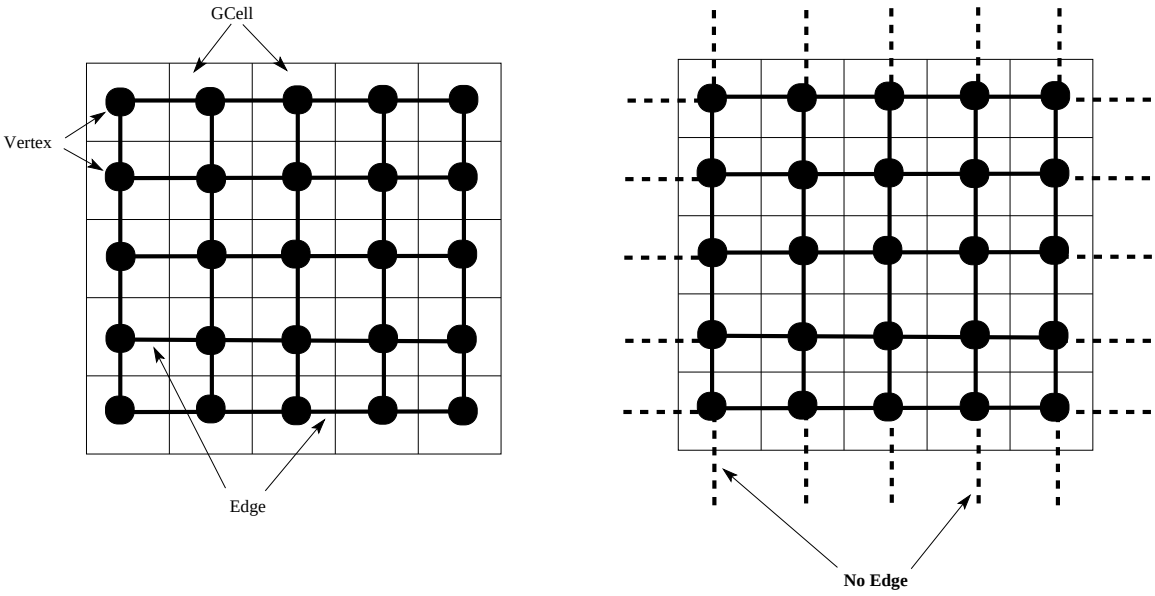


Figure 6.6: Existing Global Routing Model: (a) a partitioned layout (5x5 bins) overlaid the corresponding grid graph, and (b) boundary vertices with fewer than four edges (dotted lines)

In this model, as shown in Figure 6.6 (b), all the GCell vertices except those on the four sides of the exterior boundary of the chip layout have four edges with their adjacent GCell vertices. Those GCells at the chip boundary share their boundary with up to three adjacent GCells and the outlined layout; those GCells are either situated at the four corners or along the left, right, bottom and the top boundary of the chip layout. Practically, no routing is possible crossing the layout boundary into one of these GCells by the existing global routing methods. Moreover, as mentioned earlier, no local (bin) routing from the bin center to the pin of a net in any of the

GCells, including those along the four boundary regions, is allowed in this global routing framework and move to detailed routing stage instead.

In the grid graph model, the *routing demand* through an edge e, denoted as d_e, is defined by the number of routing tracks used by a set of nets passing through that edge. It consists of either horizontal tracks through vertical edge boundary or vertical tracks at the horizontal edge boundary between two adjacent GCells. The maximum number of vertical (horizontal) tracks crossing the corresponding horizontal (vertical) GCell boundary is called *routing capacity* r_e of the edge e. The effective capacity of each edge is computed based on the routing blockages along that edge. For example, if r_e be the routing capacity of edge e without any blockage consideration and b_e be the blockage along that edge, then the effective number of tracks that can be used to route the nets through any metal layer is given by r_e - b_e. This is also called as the effective routing capacity r'_e. At any given routing instance, for the *routing demand* d_e, the congestion in edge e is measured in terms of *overflow* as:

$$ovf(e) = \begin{cases} d_e - r'_e & \text{if } d_e > r'_e \\ 0, & \text{otherwise} \end{cases}$$

Typically, the number of GCells (hence the value of m) in a layout is predefined based on the layout area and determines the area of each GCell. If the value of m is too high, implying smaller GCell size, the global router yields more accurate routing results, but takes more computation time in order to identify the routing path of a net through these GCells. On the other hand, smaller m values yield faster routing completion, but with inferior (less) routing solutions due to larger GCell size. In this global routing framework, the routing path of a net is identified between a pair of GCells containing the pins of a net, from the center of one GCell to another GCell. Therefore, an exact pin-to-pin routing path of a two terminal net is not feasible in this framework, even if the size of the GCells is substantially reduced. Practically, this is not possible to route the local segments (center of a GCell to the pin) of all these nets, but theoretically possible with zero sized GCells, i.e., m is equal to infinity. Therefore, this is a theoretical limitation of the grid graph model and has direct impact on a well known global routing problem known as *pin access problem*. Few instances of this pin access problem are illustrated in Figure 6.7, with dotted lines highlighting the local routes to the pins from the bin center that are not completed by the existing global routers [22, 26, 28, 29, 45, 119].

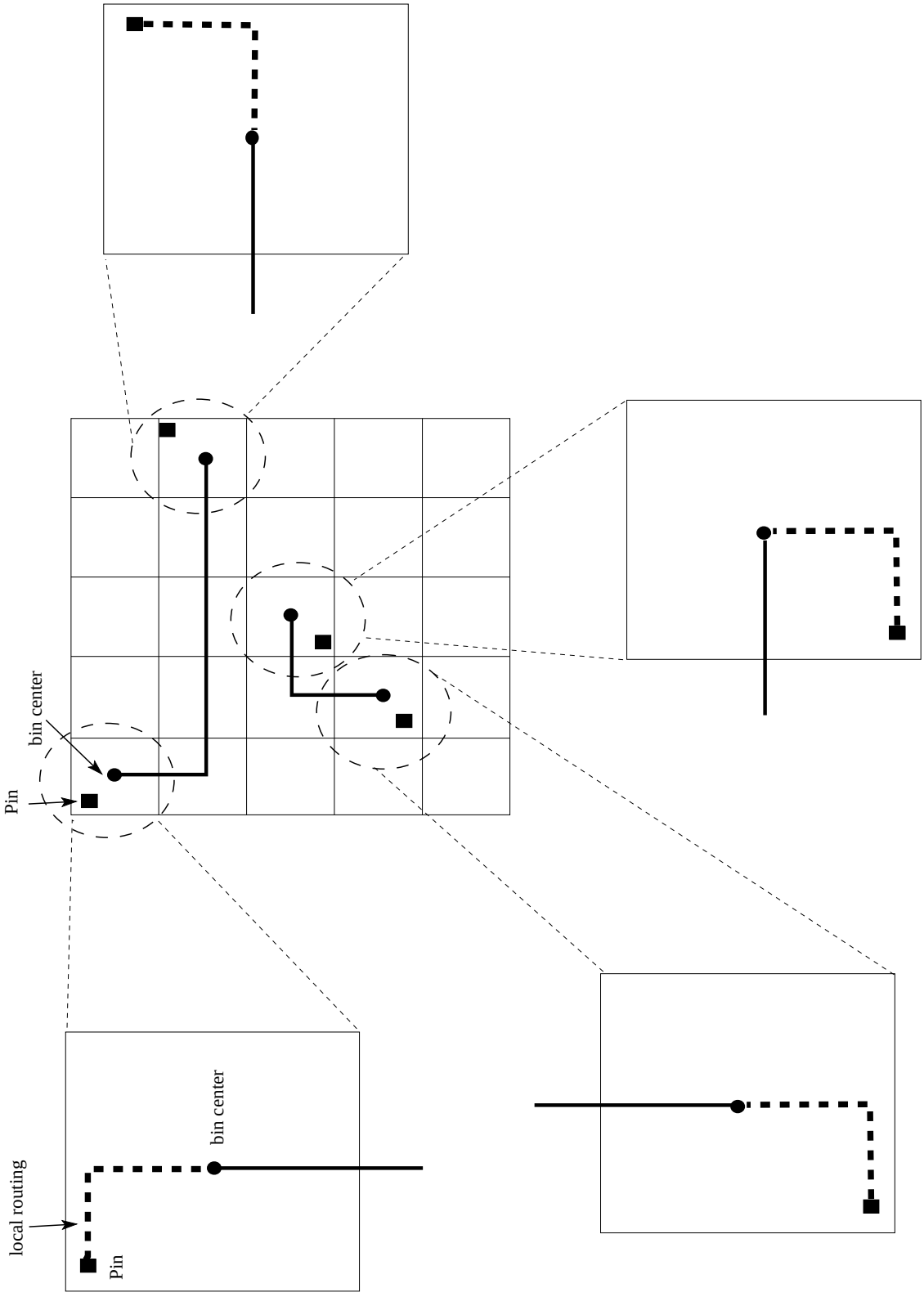


Figure 6.7: Existing global routing framework and Intra-bin (local) routing

A special case of this pin access problem arises when both the pins of a two pin net fall within the same GCell and hence do not cross GCell boundaries. These kinds of nets are not even considered during the existing global routing approaches. This problem is equally applicable for a net with more than two pin nets. In fact, the local routing of a net segment from bin center to a pin is passed on to the subsequent detailed routing stage. This is illustrated in Figure 6.7 for two different nets and their routing path from one bin center to another containing their respective pins. In this figure, the dotted lines in the zoomed-in GCells denote the local routing not performed during the traditional global routing. This has significant impact on the effectiveness of these methods in terms of wirelength, inaccurate estimation of local congestion within a routing bin (GCell) and the number of vias required for these local routes. In all practical cases, routability is greatly affected due to the local congestion scenario within the GCells, whereas the vias occupy significant amount of routing resources (tracks). Therefore, multiple iterations are necessary during both global and detailed routing. Several interleaved global and detailed routing methods such as [45, 46, 48, 49, 120, 121] were proposed in order to address this problem in multiple iterations of both routing methods, using incremental improvement of the combined routing solution.

6.3.1 Background

In Chapter 4, we presented the first of its kind early global routing framework STAIRoute using floorplan bipartitioning results obtained by the recursive methods such as those presented in Chapters 3 and 5. The resulting bipartitions on a given floorplan identify a set of monotone staircases as the routing regions through which the routing path of a net is identified through multiple metal layers. The routing capacity of such a region is obtained from the net cut information of the corresponding node in the bipartitioning hierarchy (MSC tree). We have also shown that the topology of these routing regions and their corresponding routing capacity vary depending on the trade off parameters used in these heuristic bipartitioning methods. This also shows the impact of these trade-off parameters pertaining to the number of bends in a monotone staircase region on the number of vias along the routing path of a net through a number of metal layers. In STAIRoute, we attempted to address the pin access problem at the floorplan level, by suitably defining a set of edges in the corresponding routing graph (GSRG). Each of these edges, called pin-junction edges,

connects a pin of a block to a T-Junction in the floorplan. Therefore, STAIRoute always obtains a routing path of a net that originates from a pin and terminates on another.

Notably, STAIRoute restricts the early global routes of the nets through these monotone staircase routing regions only. These regions are located in the adjoining boundary regions of the blocks, both macros and soft blocks, in the floorplan using a number of specified metal layers. In practical designs, a macro block usually occupies some, if not all, of those metal layers for routing the nets internal to it; starting from the bottom most metal layer say M_1 up to a layer say M_j below the maximum permissible metal layer M_{max}. Therefore, the routing layers $\{M_1 \cdots M_j\}$ in the regions over these macros are blocked for routing the nets, while $\{M_{j+1} \cdots M_{max}\}$ layers are available using preferred (horizontal/vertical) routing directions. The same is also applicable to soft blocks for routing its internal nets, apparently blocking a number of metal layers say $\{M_1 \cdots M_k\}$ $(M_k < M_{max})$ for floorplan level intra-block routing. In STAIRoute, these routing blockages are assumed to be present in all available metal layers, by restricting the routing through the monotone staircases only. These leaves no metal layer available for routing the nets over these blocks, i.e. no *over-the-block* routing is permitted. This routing framework is thought to be reasonable as there are fewer number of nets at the floorplan abstraction level comprising of macros and soft blocks than the original flat netlist seen at the top level. Hence, the routing of the nets contained within the soft blocks connecting the standard cells is beyond the scope of the entire thesis work.

6.3.2 The Hybrid Routing model

In this section, we consider a scenario when a macro (or soft block) allows the global routing paths of the nets over it beyond some specified metal layer say M_j, using layer $M_j + 1$ up to M_{max} used for fabrication. In this routing framework, the routing up to M_j is done through the monotone staircase regions similar to STAIRoute i.e. in $\{M_1 \cdots M_j\}$ layers while the grid graph model [17] is used for obtaining over-the-block routing in the subsequent layers $\{M_{j+1} \cdots M_{max}\}$. In this work, we adopt a slightly different variant of the existing grid graph model along with the junction graph model presented in the previous chapter. The grid graph is overlaid on the junction graph to form a hybrid graph model. Alike the global staircase routing graph (GSRG) obtained by augmenting the junction graph for each net in the case of STAIRoute,

this multi-layer hybrid graph model is also augmented for each net and a shortest routing path is obtained through multiple metal layers using reserved layer model. We start our discussion on the adoption of the existing grid graph model, followed by the congestion model and the construction of the hybrid routing graph for each net.

6.3.2.1 The Grid Graph Model

In the existing grid graph based routing model, the entire layout is divided into m-by-m global routing tiles (GCells), where m is predefined. In our work, we obtain the value of m based on the number of blocks in it, there by removing the dependency on the layout (hence GCell) area and the routing efficiency. The value of m is computed as the ceiling of the square root of the number of T-junctions i.e. $\lceil\sqrt{(2n-2)}\rceil$, where n is the number of blocks and $2n-2$ is the number of T-junctions in the floorplan [105] (also see Lemma 6).

The grid graph $G_g = (V_g, E_g)$ is defined as follows: each GCell corresponds to a vertex $v_p \in V_g$ while each edge $e_{pq} \in E_g$ denotes a pair of vertices (v_p,v_q) such that the GCells (g_p,g_q) corresponding to v_p and v_q share a common boundary. Notably, the number of vertices $|V_g|$ and edges $|E_g|$ can be obtained as m^2 and $2m(m-1)$ respectively. Hence, both these parameters depend solely on the total number of blocks (macros or soft blocks) n in a given design, not on the floorplan topology.

Lemma 13 *Given a floorplan with n blocks, the grid graph G_g can be constructed in $O(n)$ time.*

Proof It is evident from Figure 6.6 that there are $O(m^2)$ vertices and $O(m^2)$ edges in the grid graph G_g. Hence, its construction takes $O(m^2)$ i.e. $O(n)$ time. □

As discussed earlier, the edge capacity in the planar grid graph model is obtained based on the planar routing blockage in the lowest routing layer pair (M_1, M_2). They are projected on the routing layers beyond M_2 based on the technology defined routing track pitch and metal width. In our version of grid graph model, the routing capacity of each edge e_{pq} is computed based on the floorplan bipartitioning resuls and also the fact that if the corresponding boundary between the designated pair of tiles is fully or partially contained within the bounding box of a net n_i, it accounts for a capacity of 1. This is due to the fact that the net can take have a potential routing path through any of these GCells. In this way, the capacity of all the edges for all the nets $N = \{n_i\}$ are computed before the routing process starts.

6.3.2.2 The Congestion Model

We discuss the junction graph $G_j = (V_j, E_j)$ defined in Chapter 4 and also in [112] as below:
$V_j = \{J_p\}$, corresponds to a set of T-junctions, and
$E_j = \{\{J_p, J_q\} \mid$ a pair of adjacent junctions $\{J_p, J_q\}$ with a vertical/horizontal segment s_k of a monotone staircase $C_m \in \mathcal{C}$ between them$\}$.

The weight of each edge $e_{pq} \in E_j$ is computed as:

$$wt(e_{pq}) = length(s_k)/(1 - p_{s_k}) \tag{6.2}$$

where p_{s_k}, the *congestion* through the segment s_k between $\{J_p, J_q\}$, is defined as:

$$p_{s_k} = u_{s_k}/r_{s_k} \tag{6.3}$$

where $(1 - p_{s_k})$ is defined as the *congestion penalty* on the edge weight for routing a net through this edge. As stated before, the reference capacity r_{s_k} for a rectilinear staircase segment is computed from the net cut information of the bipartitioning results. Before routing, u_{s_k} is set to 0 and if a net n_i is routed through the corresponding segment s_k, then u_{s_k} is incremented by 1. This penalty remains the same for other edges in the routing graph (GSRG) that are paired by terminals and junctions (refer to Chapter 4 for details).

The same routing penalty as per Equation 6.2 is applied on the edges in this grid graph model. The corresponding length parameter for an edge e_{pq} between a pair of adjacent GCells (g_p,g_q) is denoted as $length_(e_{pq})$ and signifies the distance between the center of the GCell pair (g_p,g_q).

6.3.2.3 The Hybrid Global Staircase Routing Graph

In the proposed hybrid routing graph model [122], both the junction graph G_j and the adopted grid graph G_g act as the backbone graphs. We call it as *hybrid global staircase routing graph* (hGSRG) $G_{ri} = (V_{ri}, E_{ri})$ for a given net n_i, similar to the routing graph (GSRG) defined in Chapter 4. For a given net $n_i \in N$ each having t_i pins, G_{ri} is defined as:
$V_{ri} = V_j \bigcup V_g \bigcup t_i$, and
$E_{ri} = E_j \bigcup E_g \bigcup E_{tjg}$
where, E_{tjg} denotes an additional set of edges between (a) a pin and a junction, (b)

a pin and a G-Cell, and (c) a junction and a G-Cell and is denoted as:
$E_{tjg} = \{t_i, J_k\} \bigcup \{J_k, g_m\} \bigcup \{t_i, g_m\}$
Here, $\{t_i, J_k\}$ denotes an edge between a net pin t_i and a T-junction J_k in lower layer group (M_1, M_2) pertaining to the junction graph G_j (see GSRG in Chapter 4); $\{t_i(J_k), g_m\}$ denotes a vertical edge between a pin t_i (junction J_k) and a G-Cell g_m such that t_i (J_k) in lower metal layer group (M_1,M_2) is located within the planar boundary of the GCell g_m. The edges $\{t_i(J_k), g_m\}$ in this hybrid routing model facilitates the local routing (pin accessibility) within the GCell. The steps for the construction of this hybrid routing graph is illustrated in Figure 6.8.

After successful routing of a net n_i, the capacity of the corresponding segments in the monotone staircases and grid edges along the routing path of n_i is updated by 1. In this work, we assume that the pins of a block are located in metal layer M_1 or M_2. The pins residing in lower metal group (M_1, M_2) are associated with the junction graph, as shown in Figure 6.9, forming pin-junction edges similar to the graph model used in STAIRoute. The pin-GCell and junction-GCell edges in G_{ri} are constructed by identifying the pins/junctions within the corresponding GCell, by overlaying the grid graph on the junction graph. The routing cost of these edges, being the vertical connecting edges between G_j and G_g, is simply the planner length between the pin/junction location and the center of the GCell and the number of vias incurred due to routing through multiple layer groups. Alike the existing post-placement global routers, no capacity constraints for these edges are considered as congestion through these vertical edges has little significance, except via overhead.

The main contribution in this work is that we use G_j for identifying the (partial) routing path of a net in lower group of metal layers such as M_1-M_2 only, while G_g is used to route the nets in the subsequent higher metal layers beyond M_2. The routing through G_g takes place when a segment of a net cannot be completed in M_1-M_2 layers, due to the congestion restriction in the routing regions, as illustrated in the subsequent section.

6.3.2.4 Illustration of Hybrid Routing

We illustrate the working of this early global routing framework in the Figure 6.9 using the routing instances of a 2-terminal net (t_a, t_j) in the given floorplan instance. The first example in Figure 6.9 (a) shows that the routing path of the net is entirely confined within the monotone staircases in lower metal layers only, e.g., in (M_1,M_2)

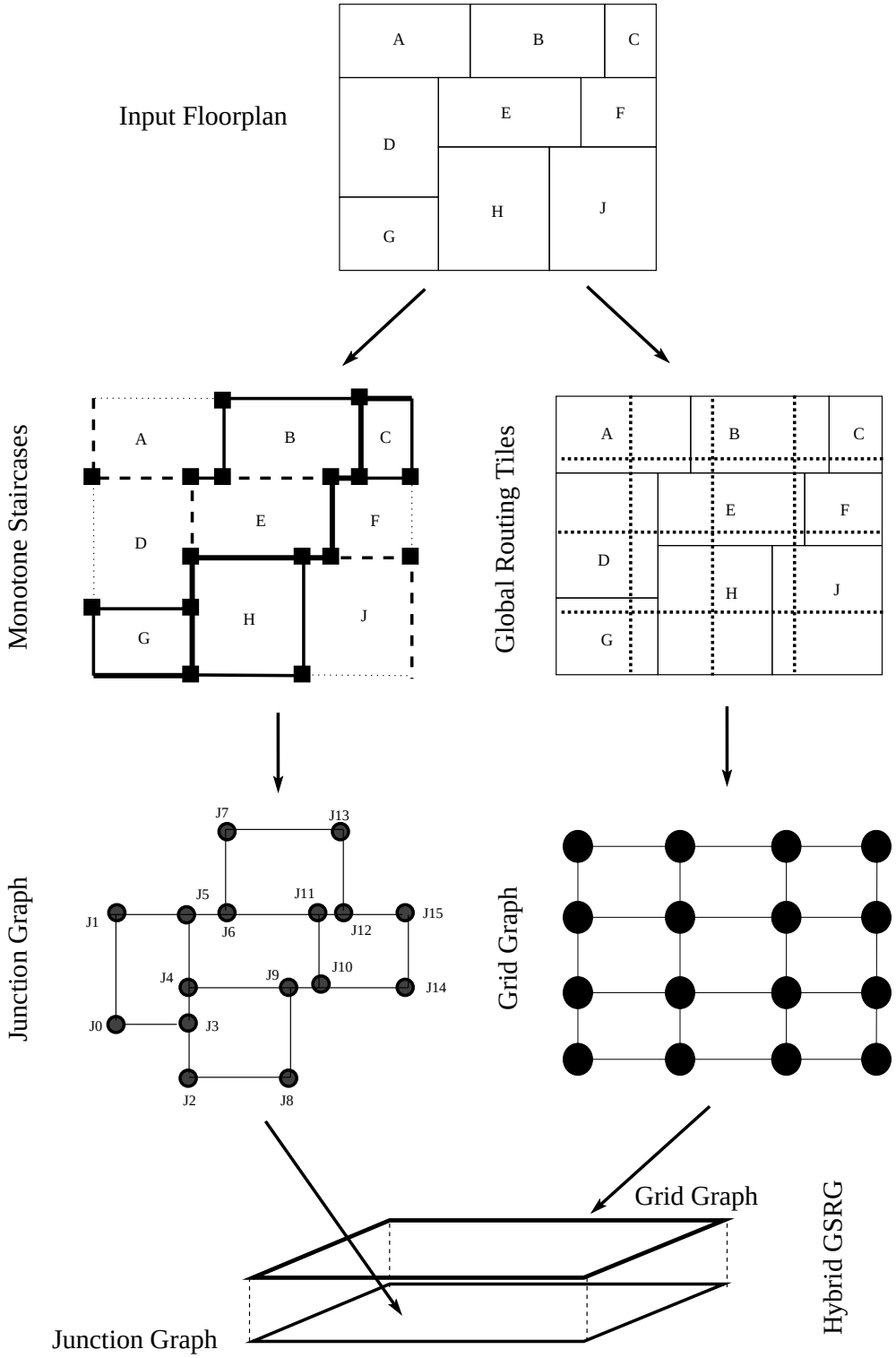


Figure 6.8: Construction of hybrid global staircase routing graph (hGSRG)

layer pair. This is similar to the routing path obtained by STAIRoute presented in Chapter 4. The second example in Figure 6.9 (b) illustrates another routing instance where the routing path is not fully confined to the monotone staircases, between J_2 and t_j passing through J_3, due to congestion in some of them (highlighted by the dotted lines) in (M_1,M_2) layer pair. Alternatively, the proposed over-the-block routing method identifies the remaining routing path ($J_2 \rightsquigarrow t_j$) through the GCells in upper metal layers, i.e., beyond M_2, by using the free space over the block H.

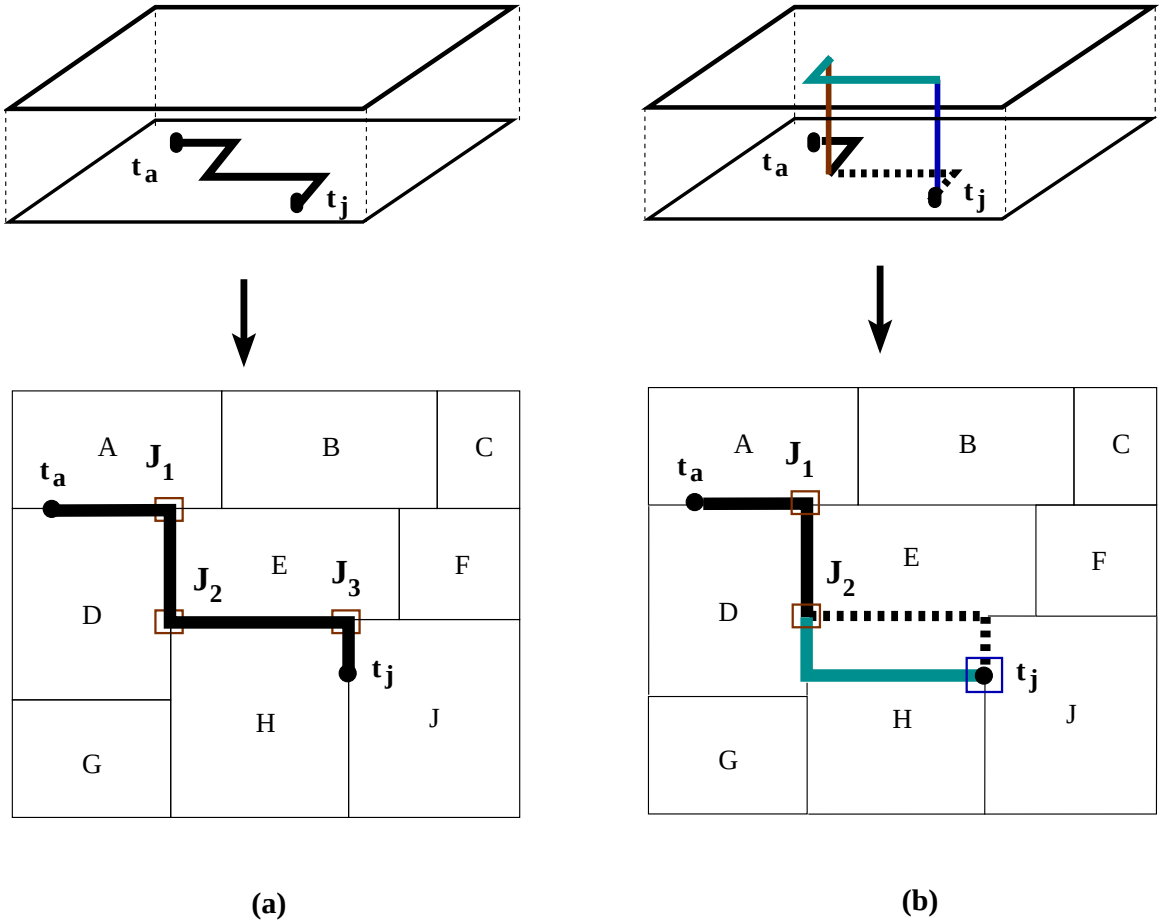


Figure 6.9: Routing instances of a 2-terminal net (t_a, t_j) using: (a) monotone staircases only in lower layer pair (M_1,M_2) by STAIRoute, and (b) the proposed over-the-block (over H block) routing

Notably, the second routing instance, as depicted in Figure 6.9 (b), comprises of three different routing steps, (i) through the monotone staircase routing regions in (M_1,M_2) from pin t_a to junction J_2 through J_1, (ii) routing in M_3 and above using GCell-GCell routing using grid graph model, and (iii) the local routing of J_2 with

one GCell center and t_j with the other GCell center in M_3 and above. These steps are illustrated in Figure 6.10 (b)-(d) respectively. For multi-terminal nets, we use the multi-terminal net decomposition method similar to that proposed for STAIRoute in Chapter 4. For each valid terminal pair i.e. the edge in the resulting spanning tree, we apply this two terminal hybrid routing method, followed by the Steiner point identification.

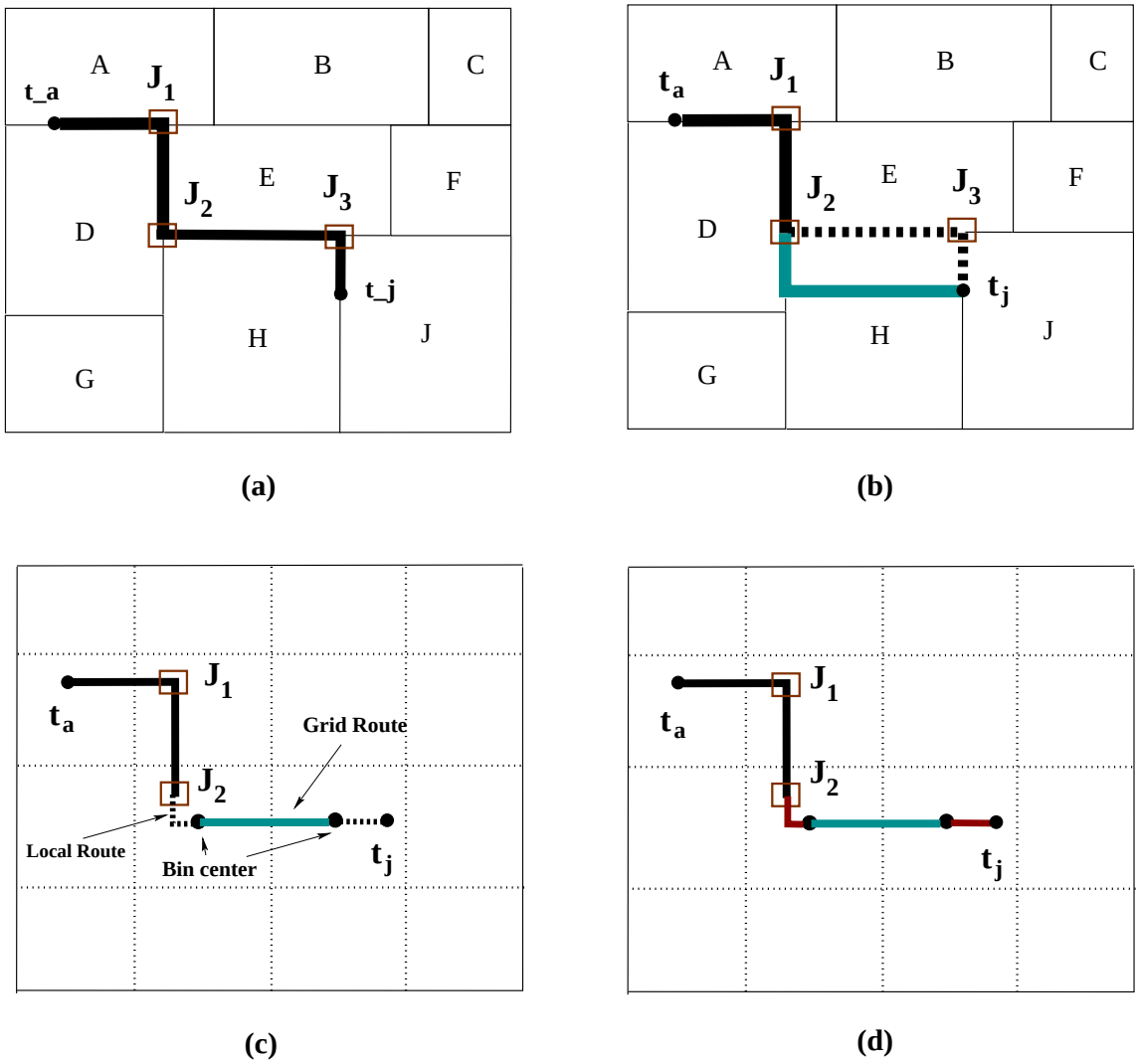


Figure 6.10: Steps for over-the-block (over H block) routing of a 2-terminal net (t_a,t_j): using (a) monotone staircase only routing by STAIRoute, (b) monotone staircases in (M_1,M_2), and grid edges in M_3 and above, (c) local routing between the pin t_j and the T-junction J_2 and the bin centers, and (d) final over-the-block (over H block) routing

In Figure 6.11, we illustrate different instances of local routing within a GCell

(bin) performed by the proposed hybrid router using pin(Junction)-GCell edges in the hybrid global staircase routing graph (hGSRG) presented earlier. The grid graph edges are used for routing the net segment in upper layers, say beyond M_2, from the center of one GCell to the center of another GCell. In this example, we focus only on the instances of partial global routes entering (or leaving) the GCell from one of its boundary edges and terminating (originating) at its center. The corresponding local routes from the GCell (bin) center to a designated pin (or T-junction) located in (M_1,M_2) falls in a particular quadrant of the bin. In this framework, the quadrant of the GCell containing the pin (junction) guides the local routes and accordingly rectify the effective wirelength of the local route needed to connect the pin (junction). Notably, these local routes use L shaped patterns and use odd layers for horizontal and even layers for vertical segment routing.

Another important aspect that this framework handles is the routing demand within a GCell, i.e., local congestion, computation based on the proposed local resource reservation (similar to track reservation). Since the routing capacity and demand are the parameters related to GCell boundary edges, these local routes are not entitled to utilize them as per the existing grid graph model. On the other hand, our routing model allows us to use grid graph edges for routing the nets beyond some specified routing layers such as M_3 beyond the layers used for monotone staircase routing such as (M_1,M_2) layer pair, as well as obtain the local routes within a GCell. The grid graph edges are used to route the net segments that were not routed in (M_1,M_2), but between the centers of the corresponding GCells. These net segments may either be between two junctions or between a pin and a junction. Therefore, the remaining GCell center to pin (junction) edges are used to move the net segment to upper layers (M_3 and above) with an additional via overhead. As mentioned earlier, all the local routes use L shaped pattern for minimal via overhead. This is illustrated in Figures 6.9 and 6.10.

In Figure 6.12, we illustrate the routing demand u for each boundary of a GCell after local routing of a net is performed within the GCell. This example shows that the position of the pin (junction) in one of the four quadrants of the GCell and the global route of the net terminating on the GCell center determine the reservation of routing demand u of the corresponding boundary edges; example (a) shows that unit routing demands in the top and left edge are reserved for local routing while bottom edge demand is meant for global route, (b) uses the same like in (a) but with right edge for global route. The example of (c) and (d) are special cases that are

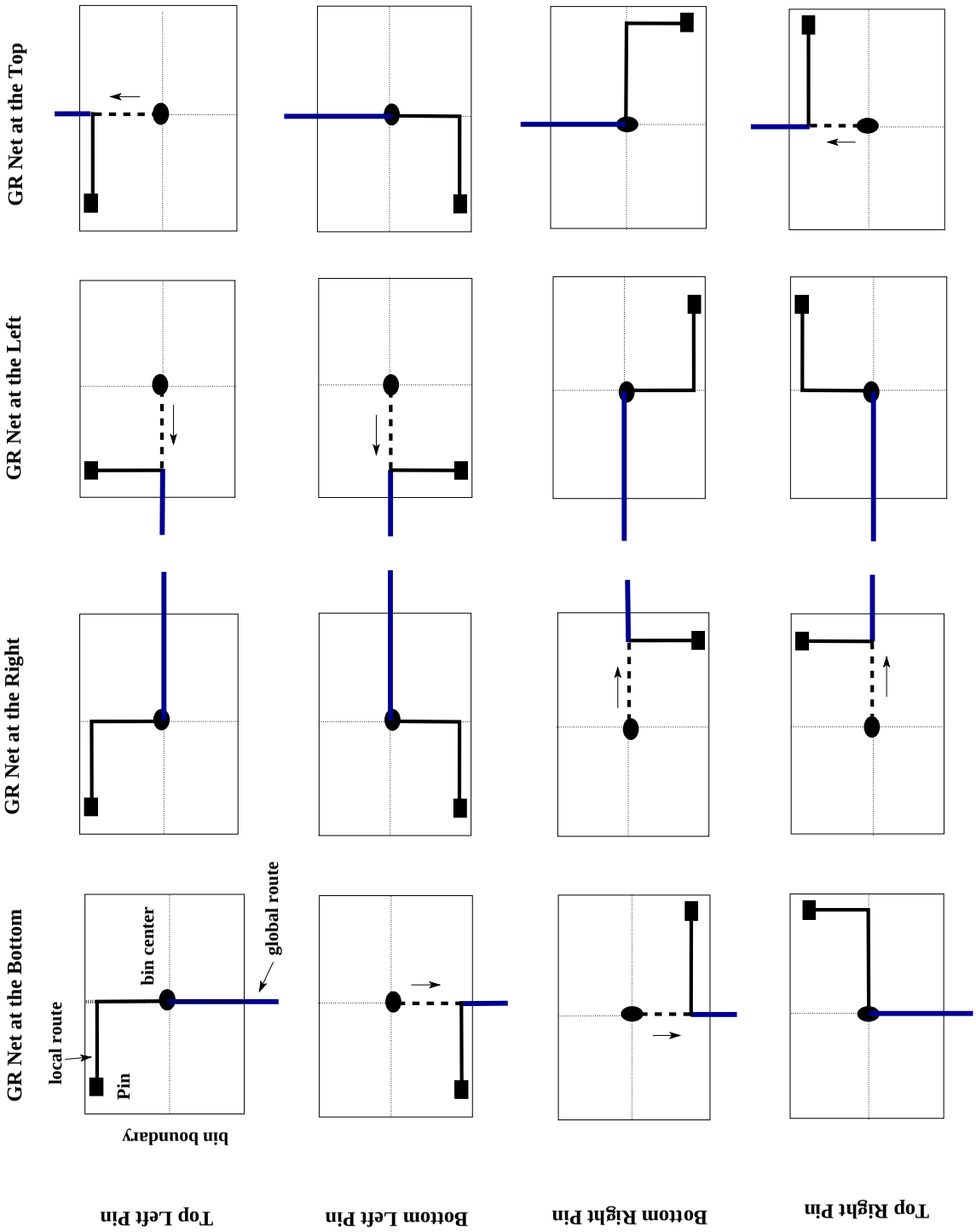


Figure 6.11: Instances of Intra-bin routing within a GCell done in this work

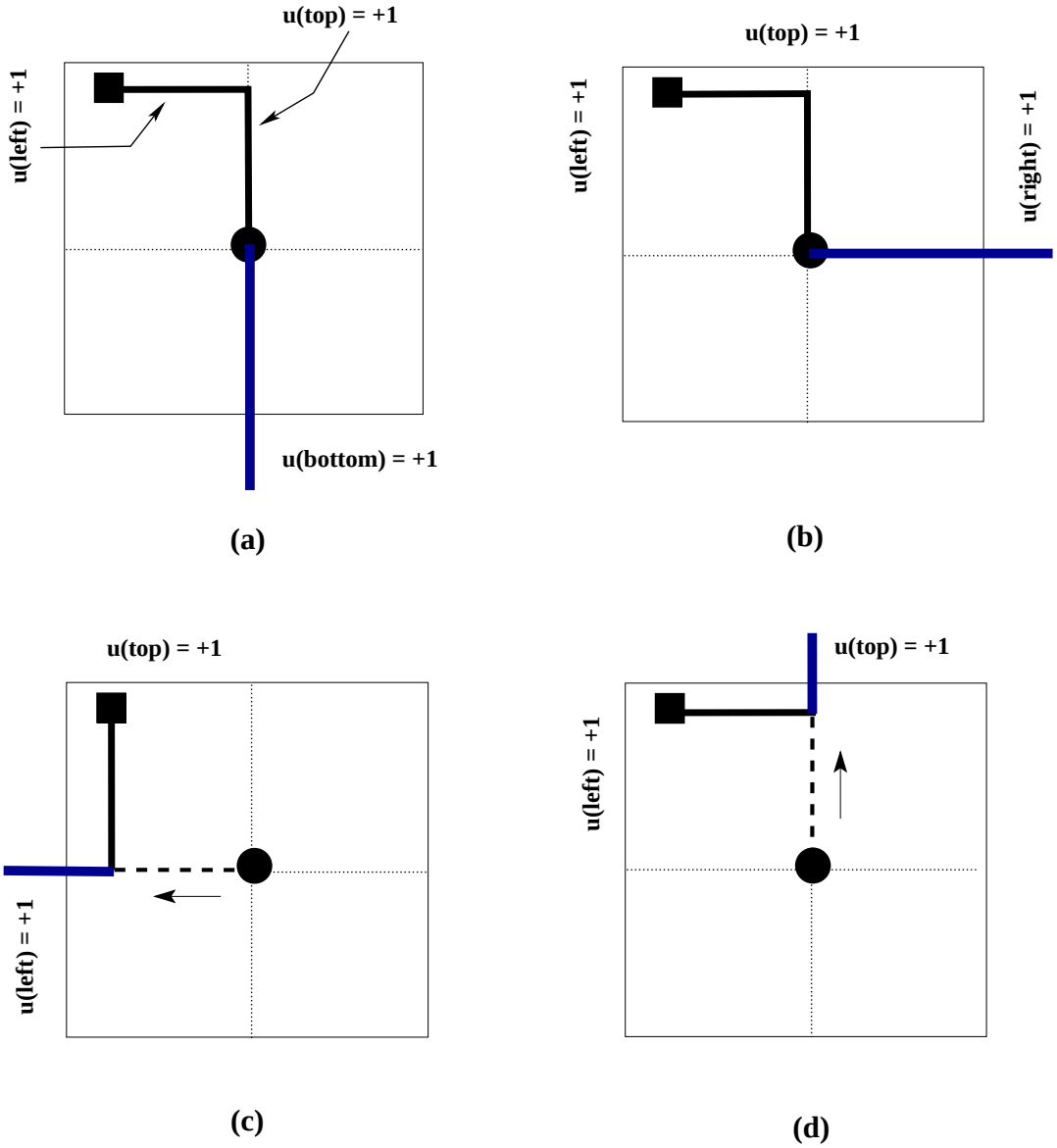


Figure 6.12: Local Routing instances using routing demand reservation of relevant boundary edges

closely related to the edge being used for global route with respect to the position of the pin (junction). While the local routing instance in (c) shows that it utilizes the same left edge capacity for both local route and the global route segments and top edge capacity for vertical demand of the local route, (d) depicts a similar case for the global route at the top edge share with vertical segment of the local route, and left edge for horizontal segment of the local route. In these cases, no other edge capacity is relevant and hence not reserved as dictated by zero values in the demand in these

edges. In this example, we also note that the wirelength is further minimized (as dictated by the arrow and the dotted line) due to common segment length between global and local routes, as is also depicted in Figure 6.11.

6.3.3 The Algorithm for Hybrid Global Routing

Algorithm 9 summarizes the steps for the proposed hybrid early global routing method *HGR*. Similar to STAIRoute, Dijkstra's shortest path algorithm [109] is used to identify the routing path of a 2-terminal net in the proposed hybrid routing graph. We use the same multi-terminal net decomposition into 2-terminal net segments for the identification of the Steiner tree topology. The set of nets N are ordered first according to net-degree, and then HPWL in the given floorplan.

Algorithm 9 HGR: An Early Global Router for Over-the-Block Routing in a Floorplan

Inputs: $G_j(V_j,E_j)$, $G_g(V_g,E_g)$, ordered nets N
Outputs: Early global routing of each t-terminal ($t \geq 2$) net $n_i \in N$ with 100% routability and Congestion $\leq$ 100%

```
for all ordered nets n_i ∈ N do
  Construct hybrid GSRG G_ri for n_i
  if n_i is a 2 terminal net then
    Identify the routing path using Dijkstra shortest path algorithm [109] on G_ri
    Compute netlength, and via-Count; update u for the respective routing resources for successful routing of n_i
  else
    Construct the node graph G_ci and identify its minimum spanning tree (MST) T_ci for this 2-terminal segment using Prim's algorithm [109]
    for all valid 2-terminal pairs (t_j, t_k) ∈ T_ci do
      Identify the routing path using Dijkstra shortest path algorithm for (t_j, t_k) on G_ri
      Compute netlength, and via-Count; update u for the respective routing resources for successful routing of this 2-terminal segment
    end for
    Identify the Steiner Point(s)
    Recompute netlength and via-count for n_i
  end if
end for
Compute congestion for all the routing resources across the metal layers.
```

Theorem 4 *Given a floorplan with n blocks and k nets each having at most t-terminals ($t \geq 2$), HGR takes $O(n^2kt)$ worst case time for finding the routing path of all the nets.*

Proof From Lemma 8 cited in Chapter 4, for a given net n_i with t pins (terminals), there are $O(t)$ pin-junctions edges. Again, for m GCells, with t pins and $2n-2$ vertices in G_j, there are $O(t)$ pin-GCell edges and $O(n)$ junction-GCell edges. For each net n_i, using Lemma 7 and Lemma 13, the hybrid GSRG G_{ri} construction takes $O(t+n)$, i.e., $O(n)$ time, since $t = o(n)$.

Similar to STAIRoute, the implementation of Dijkstra's single source shortest path algorithm takes $O(n^2)$ time. For $t(>2)$-terminal nets, both the construction of G_{ci} and finding its MST require $O(t^2)$ time (see Chapter 4). Therefore, finding a routing path for a t terminal net requires $O(n+n^2t+t^2)$, i.e., $O(n^2t)$ time. Hence HGR takes $O(n^2kt)$ time for routing k nets. □

It is evident from Theorem 4 that HGR presented in Algorithm 9 has the same time complexity as that of STAIRoute presented in Chapter 4, but with a constant time overhead due to the construction of the grid graph and identifying the pin (junction)-GCell edges. This overhead also takes into account the routing demand update in each of the edges in the hybrid routing graph hGSRG pertaining to both the junction graph and the grid graph.

It is also important to note that, as mentioned in Chapter 4, this time complexity can be improved to $O(nkt \log n)$ with a better implementation of Dijkstra's shortest path algorithm used in HGR for a general floorplan containing n blocks and k nets having a maximum of t terminals in each net. For a special case of constant average number of pins in all the nets, typically not exceeding 3/4 implying $t = O(1)$, the effort in the proposed multi-terminal net decomposition method presented in Section 4.3.2 is drastically reduced. As a result, the proposed method HGR yields a further improvement with $O(nk \log n)$ time to route the nets.

6.3.4 Early Abstraction of Edge Placement Error

In this section, we study how edge placement errors (EPE) [12, 88, 89] occur due to inefficient printability issues of sub-wavelength features using the existing optical illumination system using $193nm$ wavelength. These errors are further aggravated due to the congestion scenario in the routing regions. The intensity map in Figure

6.13 (a) depicts that the intensity is not uniform under the mask opening while the same is not zero beyond the mask opening. Therefore, it signifies additional metal width of the wire segment beyond its contour (see Figure 6.13 (b)). Thus, if a routing region is more congested, there is little scope to cope up with EPE than doing ripup and reroute for some of the nets (or a part of it), as illustrated in Figure 6.13 (b). Moreover, EPE related routing blockage to other nets may leave little room for the detailed routing of the adjacent nets. If this problem is neglected during the detailed routing stage, it will cause a failure during DFM closure stage.

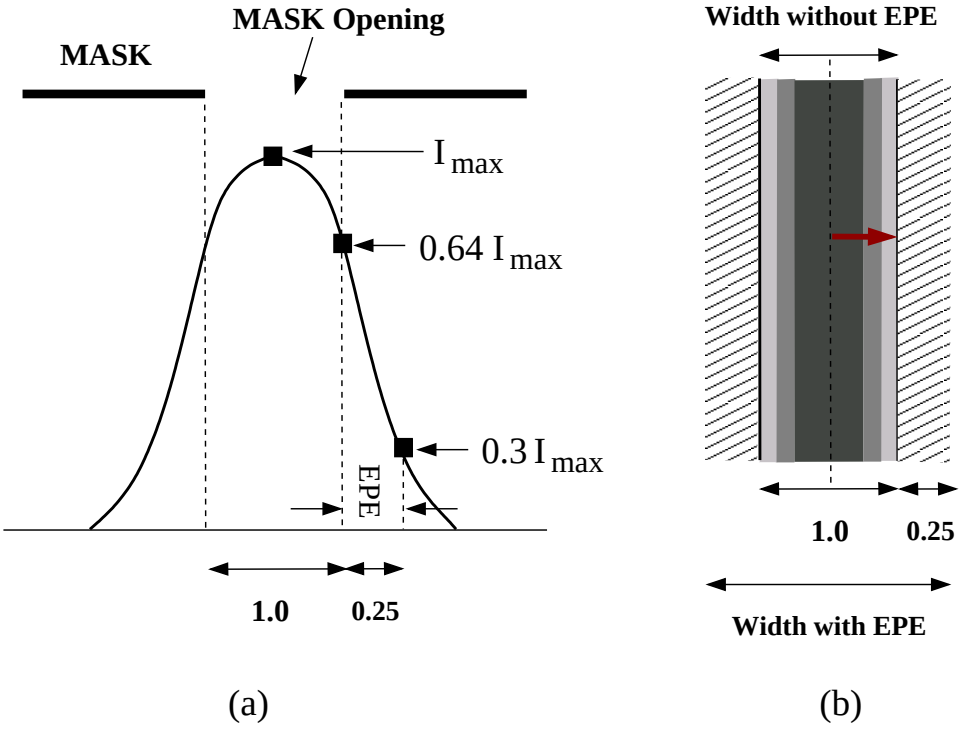


Figure 6.13: Edge Placement Error (EPE): (a) Intensity map vs. mask opening [12], and (b) actual vs. effective metal width with intensity gradient across the normalized width (dark grey to light grey from core to boundary)

The intensity map in Figure 6.13 (a) depicts the maximum intensity I_{max} at the center of the mask opening [12] and the intensity falls off gradually in a pattern similar to sinc(x) = $sin(x)/x$ function, where x is the distance measured from the center of the mask opening. Notably, the intensity at the mask boundary (edge) is around 64% of I_{max}. The intensity gradient across the width (dictated by an arrow in Figure 6.13 (b)) signifies the intensity deficiency causing optical proximity errors (OPE). In this model, we consider only the EPE effect in our early routing model in HGR due to

nonzero intensity beyond mask opening as the OPE effect due to the said intensity gradient can not be modeled without proper simulation or rule definition at this early stage of physical design flow. According to [12], we consider the threshold point for EPE as the point where the intensity falls to 30% of I_{max}, in the region beyond the mask edge, i.e., beyond the wire boundary contour. It amounts to approximately 25% increase in effective metal width on either side of the wire. As a result, it has more interfering effects on the neighboring wire segments of other nets, called EPE induced routing blockage, due to the effect of positive optical interference. This kind of violation due to EPE is discovered during optical rule check (ORC) in the physical verification process of the existing physical design flow. In this case, either wire spreading or rip-up and re-route methods are applied in order to minimize number of such violations.

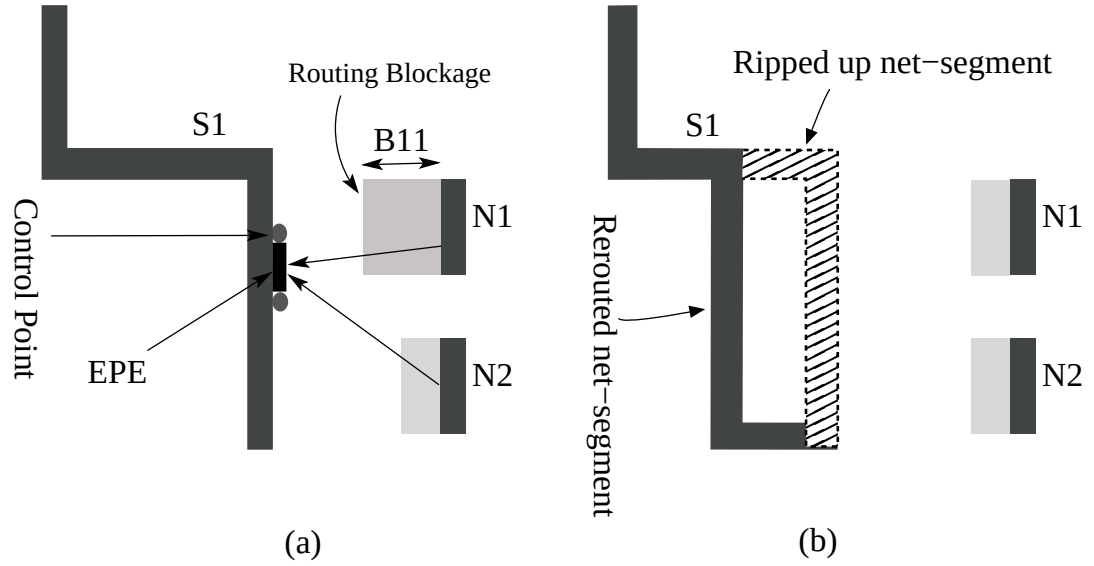


Figure 6.14: Routing blockage due to edge placement error (EPE) [12] and Rip-Reroute to alleviate it

In order to incorporate this EPE effect in our early global routing framework HGR, we abstract this effect in the routing demand for assessing the congestion scenario in any metal layer at any given routing instance. This early abstraction of EPE into our proposed routing framework HGR has the potential to reduce the lithography hot-spots due to EPE routing blockage (see Figure 6.14 (a)) at the smaller technology nodes after the detailed routing stage. Therefore, it will reduce the potential overhead of multiple iterations due to wire-spreading or ripup and reroute (RR) during detailed routing (see Figure 6.14 (b)) for EPE hotspor reduction [12, 88, 89]. In the congestion

model of HGR, the penalty due to this EPE cost abstraction is incorporated as follows: after each net is routed through the routing region e, its routing demand u_e is incremented by 1.5 considering the effect of additional 25% metal on the either side. After routing a net n_{i-1}, the congestion in the corresponding routing resources along its routing path are computed. The routing graph (hGSRG) G_{ri} for the subsequent net n_i is constructed in order to identify the routing path for it along with the present layer wise congestion scenario in the routing regions.

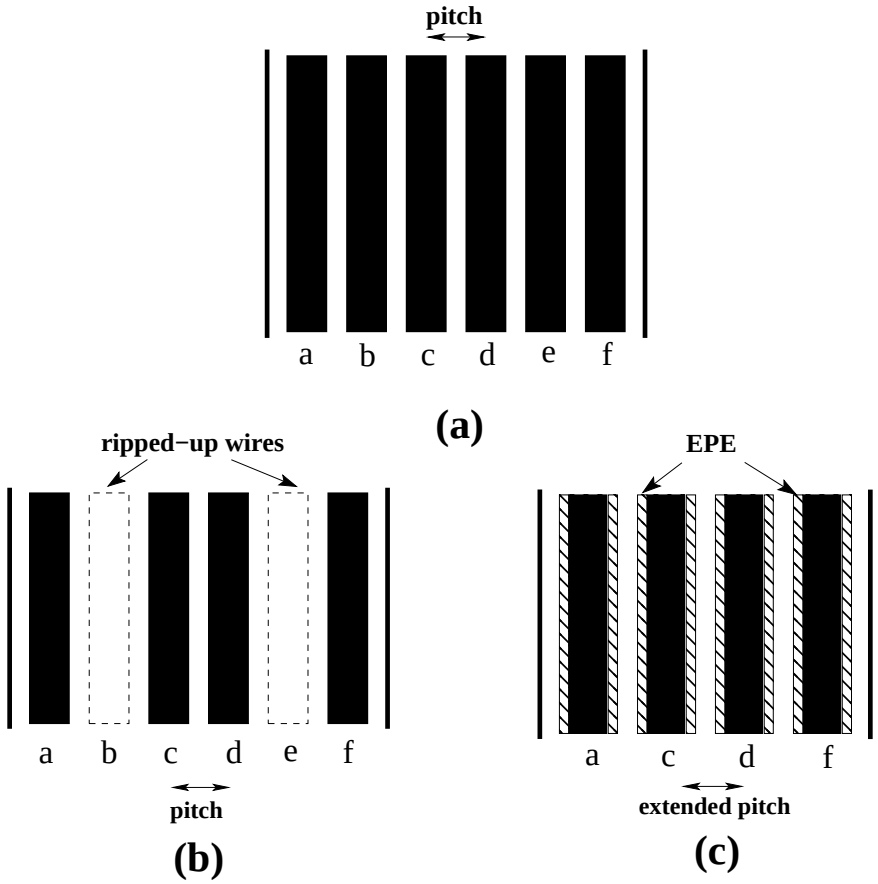


Figure 6.15: Wire spacing/pitch modulation for EPE aware early global routing: (a) all 6 tracks used, (b) 2 wires ripped up due to EPE hotspot, and (c) result of EPE aware early global routing

Due to this process, there will potentially be fewer nets to be ripped up aimed at EPE hotspot reduction during post-routing/layout optimization. The example in Figure 6.15 (a) showcases this considering six tracks ($a \cdots f$) for routing the nets, without accounting for EPE effect. However, some of the nets, in tracks b and e shown in this example, may be ripped up later during EPE aware routing optimization. On

the other hand, as depicted in Figure 6.15 (c), the proposed EPE aware early routing framework with *extended pitch*, which accounts for taking u_e as 1.5 for each net routed in region e than using 1.0 when EPE is not considered, routes only those nets, such as those remained in tracks a, c, d, f, after ripping up the other two nets during the existing methods for EPE hotspot reduction. This example shows that a routing solution considering such an early model may potentially reduce the overhead of multiple iterations due to (i) first routing the nets, and (ii) then ripping some of them up in an attempt towards EPE hotspot reduction.

6.4 Experimental Results

In this chapter, we consider another set of larger floorplanning benchmarks, namely IBM HB floorplanning benchmarks [16], for verifying the proposed early global routing methods considering the DFM issues such as uniform wire distribution for minimizing layer wise topology variation due to CMP and edge placement errors (EPE). These benchmarks were derived from ISPD98 placement benchmark circuits with certain modifications. These modifications were done in order to form a set of clusters from the standard cells present in the placement benchmark circuits to create the soft blocks as well as extract a subset of the original netlist comprising of the macros and standard cells. Using these benchmarks, we verify our proposed early global routing methods WDRoute and HGR presented in this chapter in order to study the effect of uniform wire distribution and an early abstraction of EPE in the congestion model. In this chapter, we also present the results for these benchmarks obtained by STAIRoute.

Similar to the benchmarks used in the earlier chapters, the floorplan instances were generated using *Parquet* fixed outline floorplanning tool [14, 113] using random seed for each of the circuits in this benchmark, i.e., *ibm*01 to *ibm*10. The algorithms proposed in this chapter, WDGRoute and HGR, were implemented in C programming language and the experiments were conducted on a Linux platform with Intel Xeon processor running at 2.4 GHz speed and has 64 GB RAM in it. Since [16] does not provide pin information, STAIRoute and HGR assumed the pins at the center of the blocks as compared to the heuristic pin distribution method around the boundary of a block necessarily aimed for WDGRoute. In these experiments, maximum eight metal layers are used to route the nets. In this section, we first present a comparison

Table 6.1: HB Floorplanning Benchmark Circuits [16]

Circuit	#Blocks	#Nets	Avg. NetDeg	HPWL ($10^6\mu$m)
ibm01	2254	3990	3.94	8.98
ibm02	3723	7393	4.84	22.19
ibm03	3227	7673	4.18	23.83
ibm04	4050	9768	3.92	30.82
ibm05	1612	7035	5.58	18.12
ibm06	1902	7045	4.92	21.78
ibm07	2848	10822	4.44	42.48
ibm08	3251	11250	4.92	46.57
ibm09	2847	10723	4.08	48.35
ibm10	3663	15590	3.85	121.23

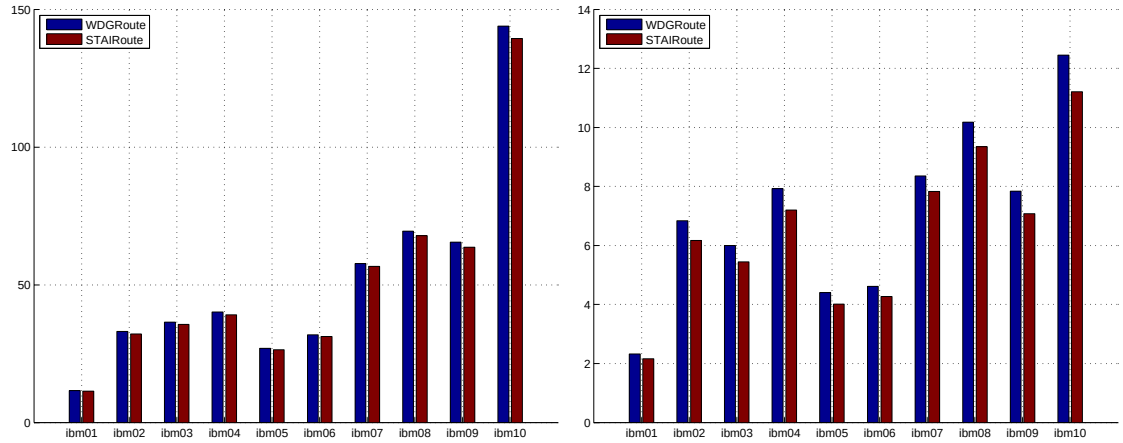
of STAIRoute and WDGRoute pertaining to uniform wire distribution in the given floorplan instances for each of the HB benchmarks. Subsequently, we study the results obtained by STAIRoute and HGR along with the effect of early abstraction of EPE in our early global routing frameworks.

6.4.1 Impact of Uniform Wire Distribution

In Chapter 4, we studied the early global routing results obtained by STAIRoute on a set of floorplanning benchmarks. However, in that study we did not emphasize on uniform wire distribution across the floorplanned layout in multiple routing layers. In this chapter, we aimed to address the impact of uniform wire distribution by a new congestion model in the proposed routing framework WDGRoute for early global routing in a floorplan. Figure 6.16 presents the results on the routing metrics such as (a) netlength, (b) via count, and (c) worst average congestion, for both STAIRoute and WDGRoute. In addition to these results, we also report the runtime needed for both the methods. Since the results in Chapter 4 were reported for smaller benchmarks, we ran STAIRoute with these new set of benchmarks for reserved layer model for up to 8 layers for fair comparison with WDGRoute.

The results show that both the methods report similar netlength for each circuit. However, the via count in case of WDGRoute increases marginally over STAIRoute as discussed before. It is to be noted that smaller wirelength implies less IR drop, and more uniformity in wire distribution correlates with less inter-layer capacitance variation due to CMP irregularities, while reduced average congestion in any layer

implies lower coupling noise due to intra-layer capacitance. All these routing results correspond to 100% routing completion and without any violation of the constraint imposed in both the congestion modes, i.e., routing demand does not cross the value of the routing capacity in any routing layer. Runtime for each benchmark circuit presented in Figure 6.16 (d) shows both the methods use similar CPU time.



(a) Netlength ($10^6 \mu m$) (b) Via count (10^5)

(c) Average worst congestion (d) CPU time (sec)

Figure 6.16: Routing results (with 100% routability) for WDGRoute and STAIRoute

In the existing post-placement global routers [22, 26, 28, 29, 45, 119], congestion in the routing edges is measured by total and maximum overflow. Instead, in Figure 6.16 (c), we present an average of the worst congestion, computed as the ratio of routing demand and routing capacity, in certain number of edges. This average congestion in certain number of edges, denoted as $ACE(x)$, is defined in [15] as follows: $ACE(x)$ in x% worst congested edges, i.e., $ACE(x)$ with $x \in \{0.5, 1, 2, 5\}$. The comparison

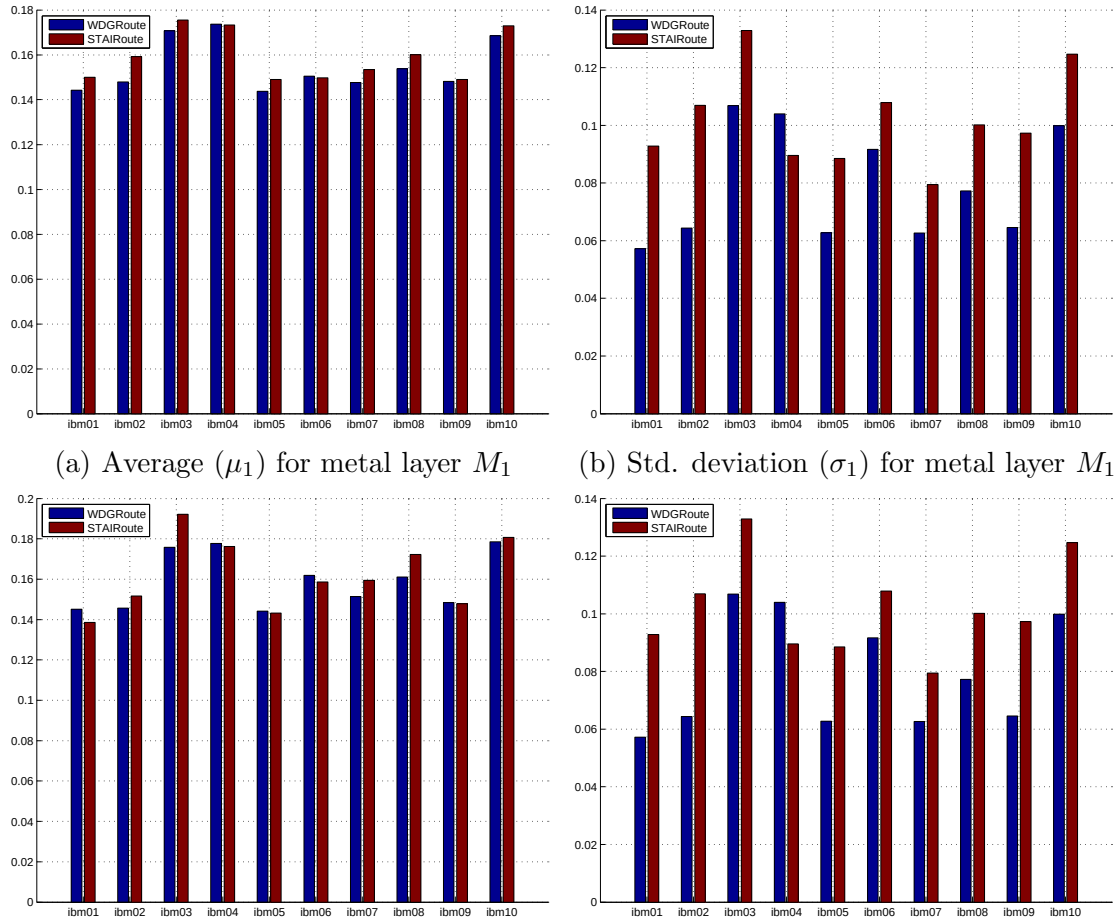
in Figure 6.16 (c), shows significant improvement in case of WDGRoute for most of the circuits with an average of 44% reduction, except for *ibm*05 and *ibm*07 circuits. These exceptions are either due to the floorplan topology or the bipartitioning results for a given trade-off that govern the routing quality.

Although the average of worst congestion gives an idea of the potential congestion hot-spots, it fails to provide the overall congestion scenario in a routing layer across the layout. Therefore, we present a study on the congestion statistics i.e. mean and standard deviation (μ, σ) of all routing edges in our routing graph model. Figure 6.17 presents a comparative study between WDGRoute and STAIRoute for the most critical routing layer pair (M_1, M_2). This study shows that WDGRoute yields improved congestion scenario for most of the circuits by uniformly distributing the nets over the routing layers, while incurring an average of 9% more vias and 2% more wirelength than STAIRoute.

6.4.2 Impact of Over-the-Block Routing

In this study, we assess the impact of both the early global routing models, one presented in Chapter 4 using monotone staircase routing regions only and the over-the-block approach for early global routing presented in this chapter, i.e., STAIRoute and HGR respectively. We first study the routing results for STAIRoute and HGR without considering early EPE cost in the congestion model. In fact, both of these methods use similar congestion model as discussed earlier in this chapter, differing only in the routing graph model. Similar experiments were also conducted for both the methods considering the EPE effect in the congestion cost while obtaining the routing paths of the nets across the routing (metal) layers. As cited before, these experiments were conducted for eight metal layers using preferred routing directions (horizontal/vertical) in odd/even layers. The corresponding results are plotted in Figures 6.18 and 6.19, without and with considering EPE cost respectively, for (a) netlength, (b) via count, (c) average worst congestion, and (d) runtime. Notably, these results were obtained ensuring 100% routing completion of the nets using up to eight routing layers.

With these experiments, our aim is study the impact of hybrid routing model in early global routing as compared to the routing model presented in Chapter 4 for STAIRoute using monotone staircases only and the same early abstraction of EPE cost in the routing penalty. The results for netlength shows almost identical



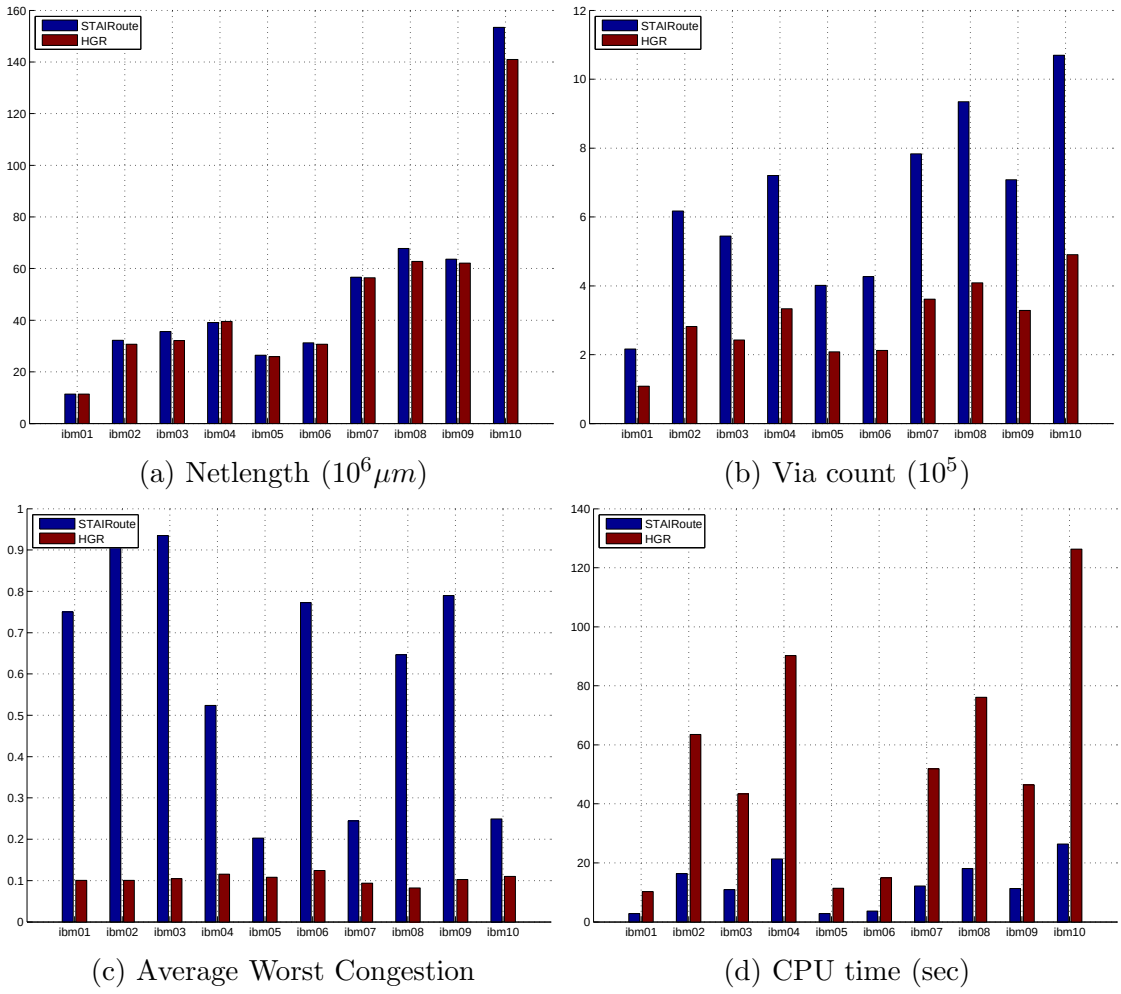
(a) Average (μ_1) for metal layer M_1 (b) Std. deviation (σ_1) for metal layer M_1

(c) Average (μ_2) for metal layer M_2 (d) Std. deviation (σ_2) for metal layer M_2

Figure 6.17: Congestion Statistics for WDGRoute and STAIRoute

values obtained by these early global routers for both without and with EPE cost in the routing penalty; HGR shows an average of 3.5% and 4.5% smaller wirelength over STAIRoute for with and without EPE cost respectively. Likewise, via count for HGR also shows significant average reduction of 53% and 54.3% respectively over STAIRoute. Similarly, the worst average congestion measured by $wACE4$ discussed previously shows an average improvement of 73% to 80% in HGR as compared to that in STAIRoute, in both the cases. However, as stated earlier in this chapter, HGR needs $4x$ runtime over STAIRoute to route the same set of nets over same set of routing layers for the same floorplan instance.

In addition to the worst average congestion values as depicted in Figure 6.18 (also in Figure 6.19) (c) and alike previous result section, we also present the congestion statistics for the most critical layer pair (M_1,M_2) for both HGR and STAIRoute on



(a) Netlength ($10^6 \mu m$)

(b) Via count (10^5)

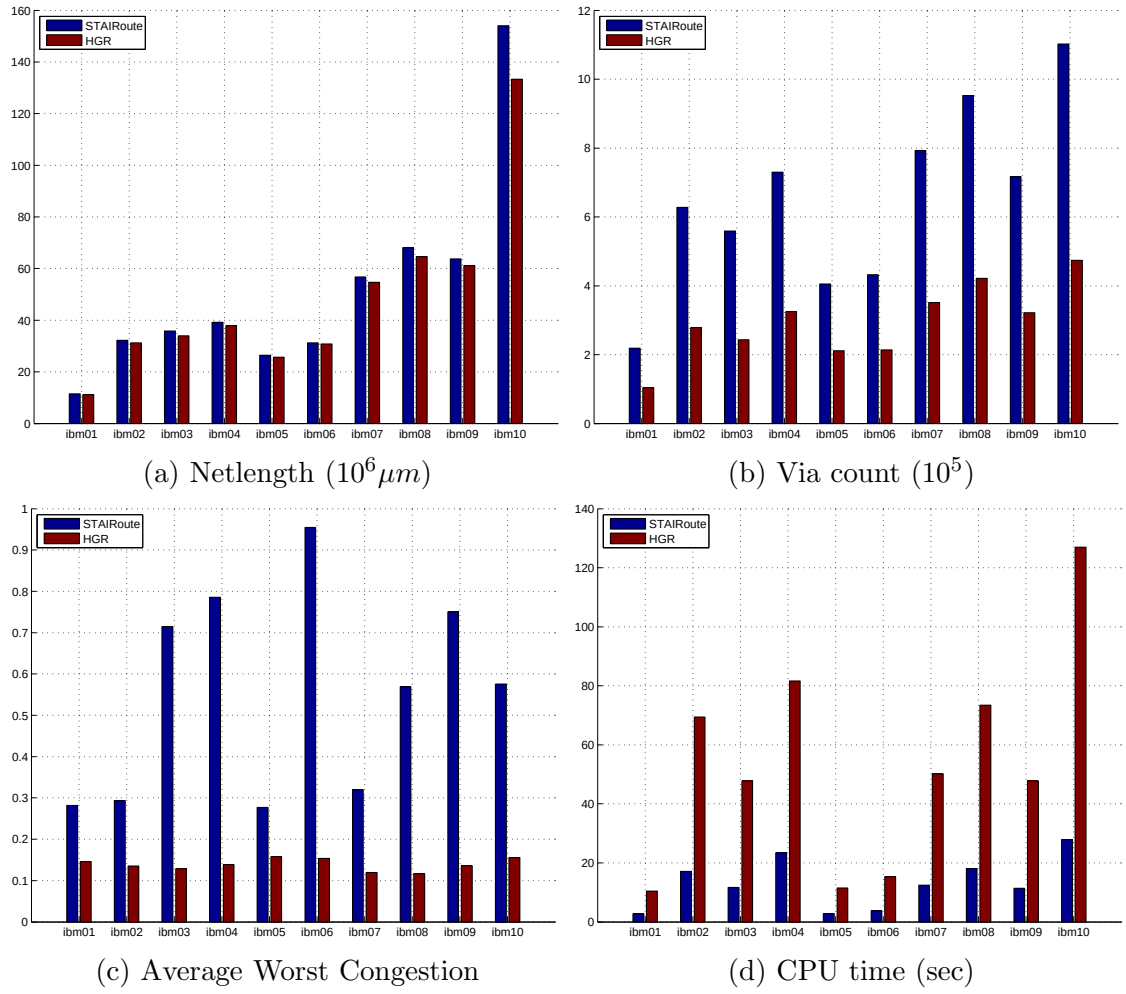
(c) Average Worst Congestion

(d) CPU time (sec)

Figure 6.18: Routing results (with 100% routability) for HGR and STAIRoute: without EPE Cost

the same floorplan instance of each of the benchmark circuits. The primary reason is that these two layers are heavily burdened with routing demand and also contain the most of the routing blockages due to placement of standard cells. In order to reflect this scenario in HGR, we use M_2 as the maximum layer being used to route all the nets within the soft blocks containing standard cells, while the layers above it are free from such global routing blockages. In Figures 6.20 and 6.21 we present the congestion statistics (μ, σ) for M_1 and M_2 routing layers, considering without and with EPE cost in the routing penalty respectively.

While M_1 shows better average congestion, 2% to 4%, M_2 is more congested in terms of the normalized geometric mean values of 9% to 18% for both without and with EPE cost. Despite these variations, the actual average congestion values in both



(a) Netlength ($10^6 \mu m$)

(b) Via count (10^5)

(c) Average Worst Congestion

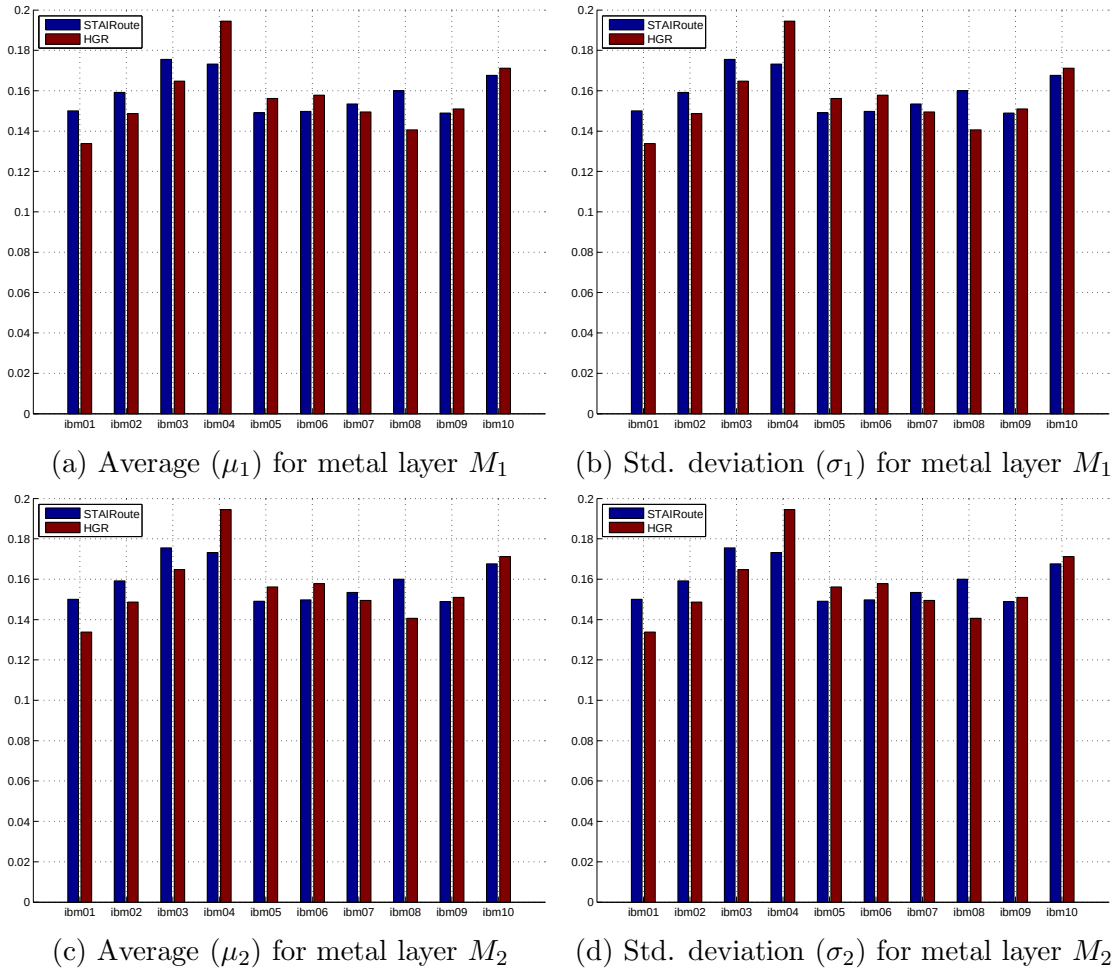
(d) CPU time (sec)

Figure 6.19: Routing results (with 100% routability) for HGR and STAIRoute: with EPE Cost

M_1 and M_2 are not so significantly different and fall within a range of [0.159, 0.189]. However, the individual congestion values for each of the circuits stay in the range of [0.124, 0.210] for all the case.

6.4.3 Comparison with Post-placement Global Routers

Since the early global routing framework presented in this thesis is applicable after floorplanning, no direct comparison is possible with the existing post-placement global routers [22, 26, 28, 29, 41, 119] for routability, wirelength, congestion, via count etc. In Table 6.2, we present a comparison of the normalized netlength for some of the popular global routing methods and the proposed early global routing methods STAIRoute



(a) Average (μ_1) for metal layer M_1

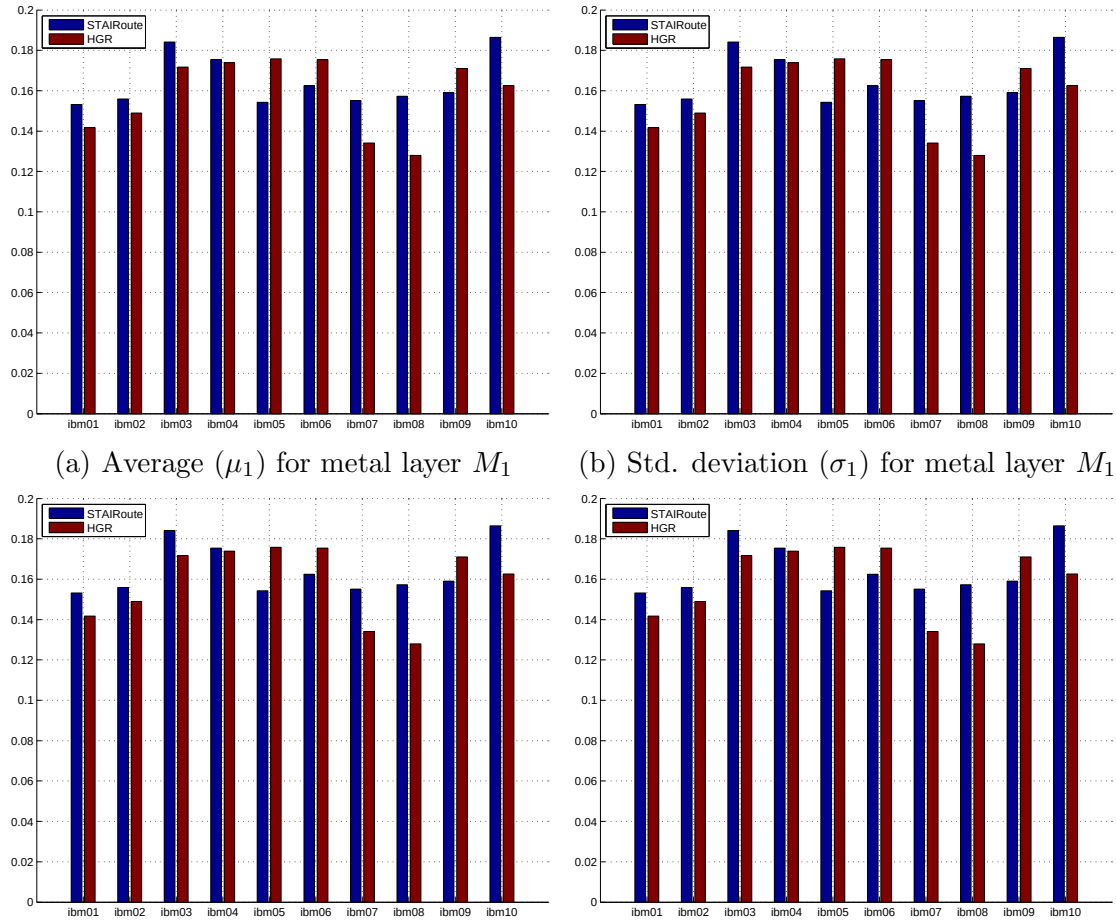
(b) Std. deviation (σ_1) for metal layer M_1

(c) Average (μ_2) for metal layer M_2

(d) Std. deviation (σ_2) for metal layer M_2

Figure 6.20: Congestion Statistics for HGR and STAIRoute: without EPE Cost

and HGR. Notably, the results reported by the existing global routers were obtained on IBM ISPD98 placement benchmark, while our methods obtained the corresponding results on IBM-HB floorplanning benchmarks. For fair comparison on the routing results, these methods need to run on the same set of benchmarks. As discussed earlier in this chapter, a floorplanning benchmark is obtained from a placement benchmark circuit using suitable clustering algorithm, the IBM-HB floorplanning benchmarks used in this work were derived from the ISPD98 placement benchmarks [16]. This abstraction incurs significant information loss due to standard cell clustering and the corresponding netlist modification. Therefore, it is unfair to compare the actual netlength obtained for the respective circuits by the existing global routers and the proposed early global routers STAIRoute and HGR. Instead, the netlength obtained for each circuit is normalized with respect to the corresponding Steiner length for all



(a) Average (μ_1) for metal layer M_1 (b) Std. deviation (σ_1) for metal layer M_1

(c) Average (μ_2) for metal layer M_2 (d) Std. deviation (σ_2) for metal layer M_2

Figure 6.21: Congestion Statistics for HGR and STAIRoute: with EPE Cost

the nets [10]. The results presented in Table 6.2 for both STAIRoute and HGR are slightly higher than those for the global routers as the nets are routed through the monotone staircase routing regions only for STAIRoute in all the routing layers, while HGR used two metal layers using these monotone staircases and rest of the layers were used in the proposed over-the-block routing.

These results show that our methods have slightly higher average values of normalized netlength than the existing global routers. As cited before, our methods restrict the routing paths for the nets through the monotone staircases only, through the lowest layer pair (M_1,M_2) for both STAIRoute and HGR. Notably, this comparison does not include any DFM costs such as uniform wire distribution and the abstracted EPE cost in the congestion model of STAIRoute or HGR.

Table 6.2: Comparing normalized (w.r.t Steiner length [10]) netlength between the existing global routers and the proposed early global routers STAIRoute and HGR

Circuit Name	Existing Post-placement Global Routers						Our Methods	
	[29]^b	[26]^b	[41]^b	[22]^b	[28]^b	[119]^b	STAIRoute^c	HGR^c
ibm01	1.071	1.042	1.068	1.053	1.059	1.039	1.156	1.147
ibm02	1.036	1.032	1.038	1.018	1.027	1.024	1.175	1.121
ibm03	1.007	1.007	1.007	1.005	1.010	1.005	1.175	1.059
ibm04	1.045	1.028	1.046	1.027	1.045	1.023	1.155	1.169
ibm05^d	-	-	-	-	-	-	1.198	1.176
ibm06	1.011	1.007	1.013	1.006	1.013	1.007	1.166	1.148
ibm07	1.018	1.006	1.015	1.007	1.016	1.007	1.192	1.187
ibm08	1.005	1.008	1.009	1.006	1.010	1.006	1.197	1.109
ibm09	1.007	1.006	1.009	1.004	1.011	1.008	1.199	1.171
ibm10	1.016	1.027	1.015	1.008	1.020	1.010	1.187	1.200
Norm. Geo Mean	0.868	0.863	0.868	0.860	0.867	0.860	1.000	0.973

b - using ISPD98 global routing benchmarks, c - using IBM-HB floorplanning benchmarks, and d - no result on $ibm05$ of ISPD98 benchmark by the existing global routers

6.5 Chapter Summary

This chapter first explored the scope of uniform wire distribution during early global routing of the nets in a floorplanned layout, intended to minimize (a) the surface irregularities arising due to CMP process due to the difference in hardness factors of the metal and the dielectric material, and (b) the congestion hotspots obtained during global routing for minimal (or no) violations in detailed routing. In this work, a new congestion model for computing the routing penality in a region is proposed based on a number of static nets as well as the number passing nets though this region, in any routing layer. Experimental results show that this model is able to route the nets more uniformly across the layout in multiple layers than the early global routing method STAIRoute proposed in Chapter 4.

Subsequently, we address over-the-block early global routing of the nets beyond some specified routing layer, since STAIRoute routes the nets in all permissible metal layers through the monotone staircase regions adjoining the block boundaries only. A new hybrid model for early global routing in floorplans, namely HGR, is presented in this chapter by adopting the existing grid graph model with suitable tuning to this routing model in order to realize over-the-block routing of the nets, when routing of a net can not be completed in lower layers through the monotone staircase regions only,

as compared to STAIRoute. In this routing framework, we subsequently incorporated an abstracted EPE cost in the routing penalty, while similar cost was also used in STAIRoute for experimental purposes. The results show that HGR yields significant reduction in via count and the worst average congestion due to the impact of over-the-block routing in the floorplan, as compared the monotone staircase only routing for all layers by STAIRoute. Congestion statistics obtained by HGR show slightly better results in M_1 than in M_2, based on the normalized mean values.

In addition to that, we also made a comparison on normalized netlength with respect to Steiner length computed with some of the existing global routers, although this is not a fair comparison. In order to evaluate the effectiveness of the proposed early global routing framework (STAIRoute/HGR) presented in the thesis, we conduct a case study with an industrial design by integrating STAIRoute/HGR in to an industrial physical design tool, presented in Chapter 7.

Chapter 7

A Case Study with Industrial Design Flow

7.1 Introduction

The existing academic global routing methods such as [22, 23, 26, 28, 29, 32, 45, 119] verified their works on the recent global routing benchmarks available online, which were derived from a relatively new placement benchmarks [123] by some of the existing placement engines [124]. However, the corresponding floorplanning benchmarks are not available for these global routing (placement) benchmarks. So far, the experimental results on the proposed early global routing methods presented in this thesis were based on the floorplan benchmarks derived from the earlier placement benchmarks [16]. This leaves little room for us to make a benchmark wise direct comparison, between the floorplan-based early global routers proposed in this work and the existing post-placement global routers. In this chapter, we conduct a case study on an industrial design run on an industrial physical design tool, by integrating the proposed early global routers.

Another important motivation of this study is that the existing global routing framework and the proposed early global routing framework below to different scope of operation as per the existing physical design (PD) flow (depicted in Figure 1.9 (a)). While these early global routers work after the floorplanning of a design without considering any information on the detailed placement of the standard cells, the existing post-placement global routers on the other hand solely depend on the detailed placement information of the cells/macros in order to compute the number

of available routing tracks at different routing layers. Moreover, the corresponding netlist information in both the cases differ due to the difference in terms of the netlist information provided. Unlike the netlist in case of the existing global routers, the floorplan level netlist does not expose the nets connected to the standard cells used in the design. Instead, a set of clusters of standard cells, commonly known as soft blocks, are obtained and are treated alike the macro blocks present in the design. Therefore, the netlist abstracted at the floorplan level do not have any information on the connectivity with the standard cells, by terminating at the boundary of the soft blocks alike the macros.

7.1.1 Outlook for This Study

The main difference between the macros and the soft blocks are that a macro block has well defined pin positions along its boundary while soft blocks lack it. Moreover, the macros have fixed aspect ratio (a ratio of width to height) as compared to the soft blocks which do not have a specific aspect ratio despite having a specific area like those macros. The floorplan optimization tools compute the aspect ratio during its optimization process and obtains a legal solution for all the blocks (macros and soft). A legal position of a block implies a feasible coordinate value of the bottom left corner that does not result in overlaps with other blocks. Subsequent placement stage optimizes the placement of the standard cells abiding by this floorplan solution with a set of clusters, aiming at the targeted *design utilization factor*. The design utilization factor is defined as the ratio of the total area of the standard cells and macros present in the synthesized verilog netlist and the total placeable area. This parameter indicates a measure on the effective space available for routing the nets in a routing layer.

Another important aspect in the existing PD flow is that the internal routing within the macros is already done using multiple metal layers, thereby no routing is allowed through those layers. However, the routing of the nets are possible over the macros above those specified layers. On the other hand, soft blocks allow routing through them in any metal layers, connecting the nets confined to the standard cells within themselves. This kind of routing is beyond the scope of the proposed early global routing framework as discussed in Chapter 6, as the interconnections defined at the floorplan level are defined among the macros, the soft blocks and the IO pads only, terminating at their respective boundary. The internal nets (from the

boundary of a soft block to the standard cells within it) are basically masked at this level of floorplanning comprising of the soft blocks, the macros and the pads [16]. The nets connecting the standard cells internal to a cluster (soft block) are unmasked during detailed placement of the cells and subsequently their routing is done using the existing post-placement global routers as cited earlier in this thesis. It is important to note that, the nets routed during the proposed early global routing are treated as a guidance for routing those (sub)nets.

7.2 Experimental Setup

In this chapter, we present the results of the case studies on two different floorplan instances of an industrial design with the help of an industry standard VLSI physical design (PD) tool called *Olympus-SoC* [13], as per the flow in Figure 1.9. This tool was provided by Mentor Graphics Corporation, a prominent industry leader in the field of VLSI design automation in very deep submicron technologies such as $65nm$ and below. In this flow, another very important tool called *Calibre-InRoute* [84] was integrated in this flow for the purpose of conducting physical verification (PV) methods such as *design rule check* (DRC) and *layout versus schematic* (LVS). These PV methods are used to evaluate the quality of the translation from a given verilog netlist into the final layout. Using this tool suite, we aim to validate the routing solutions based on a set of early global routing solutions obtained from our methods presented earlier in this thesis. As presented in the earlier chapters, these results were obtained by the proposed EGR methods, namely STAIRoute and HGR, on two different floorplan instances of the design.

According to the existing PD flow depicted in Figure 1.9 (a), the design process starts with the input structural verilog netlist, synthesized for a specific IC fabrication process node. In these study, we used $45nm$ *NanGateOpenCell* [125] library for physical synthesis and mapping of the library cells used in the given verilog netlist. Relevant process technology file was used for the physical verification steps. The aim of this case study is to observe the effect of our early global routing solutions on the final layout in terms of overall wirelength, congestion, via count, number of metal layers etc., in addition to other important metrics such as timing, cell density constrained by *design utilization factor*. The timing performance of a design implementation is a very important aspect to care for and are measured by two important

parameters called TNS (total negative slack) and WNS (worst negative slack). A timing critical design with negative TNS/WNS are not acceptable, and is considered one of the important objectives in addition to those cited above.

In Figure 7.1, we illustrate the flow setup, developed with the help of *tcl* (tool command language) framework, for the intended case study with Olympus-SoC flow [13] for physical design. In this setup, the major task was to develop a suitable interface, in order to provide a point of information access between Olympus flow and the proposed EGR framework. This interface was developed using the popular *design exchange format* (DEF) [126] for: (i) providing the required floorplan information generated by Olympus-SoC tool to the proposed early global routing (EGR) frameworks (STAIRoute/HGR) and (ii) obtaining the early global routing results obtained by the EGR tools. These results aim to guide the subsequent stages in the Olympus-SoC tool, such as detailed placement, global/detailed routing and even clock tree synthesis. In this exercise, we referred to LEF/DEF language reference manual (LRM) version 5.8 while developing the DEF interfaces that exchange the designated information in DEF format between STAIRoute/HGR and Olympus-SoC, as depicted in Figure 7.1. This includes the floorplan information for the macro and soft blocks present in the design along with their interconnection (netlist) details abstracted at the floorplan level (similar to [16]), from the floorplanning stage of the Olympus-SoC flow.

As we have seen in the earlier chapters, the EGR frameworks comprise of (a) the recursive floorplan bipartitioning methods (see Chapter 5) for identifying a set of optimal monotone staircase routing regions in the given floorplan based on certain trade-off (γ, β), and (b) the early global routing engines STAIRoute/HGR (see Chapters 4 and 6) that employ these routing regions to route the nets through multiple routing (metal) layers using preferred routing directions. The resulting early global routes of these abstracted nets are fed back to the Olympus-SoC flow using the DEF interface before the standard cell placement stage. On getting these information, Olympus-SoC tool performs subsequent optimization at each stage, i.e., clock tree synthesis, timing driven detailed placement, global routing of the nets that fully/partly remained unrouted in the EGR framework due to the limitation discussed earlier.

In this setup, we were allowed to use maximum 10 metal layers for routing all the nets and a set of (γ,β) values for the floorplan bipartitioning method used in the early global routing framework for both STAIRoute and HGR. The design utilization factor

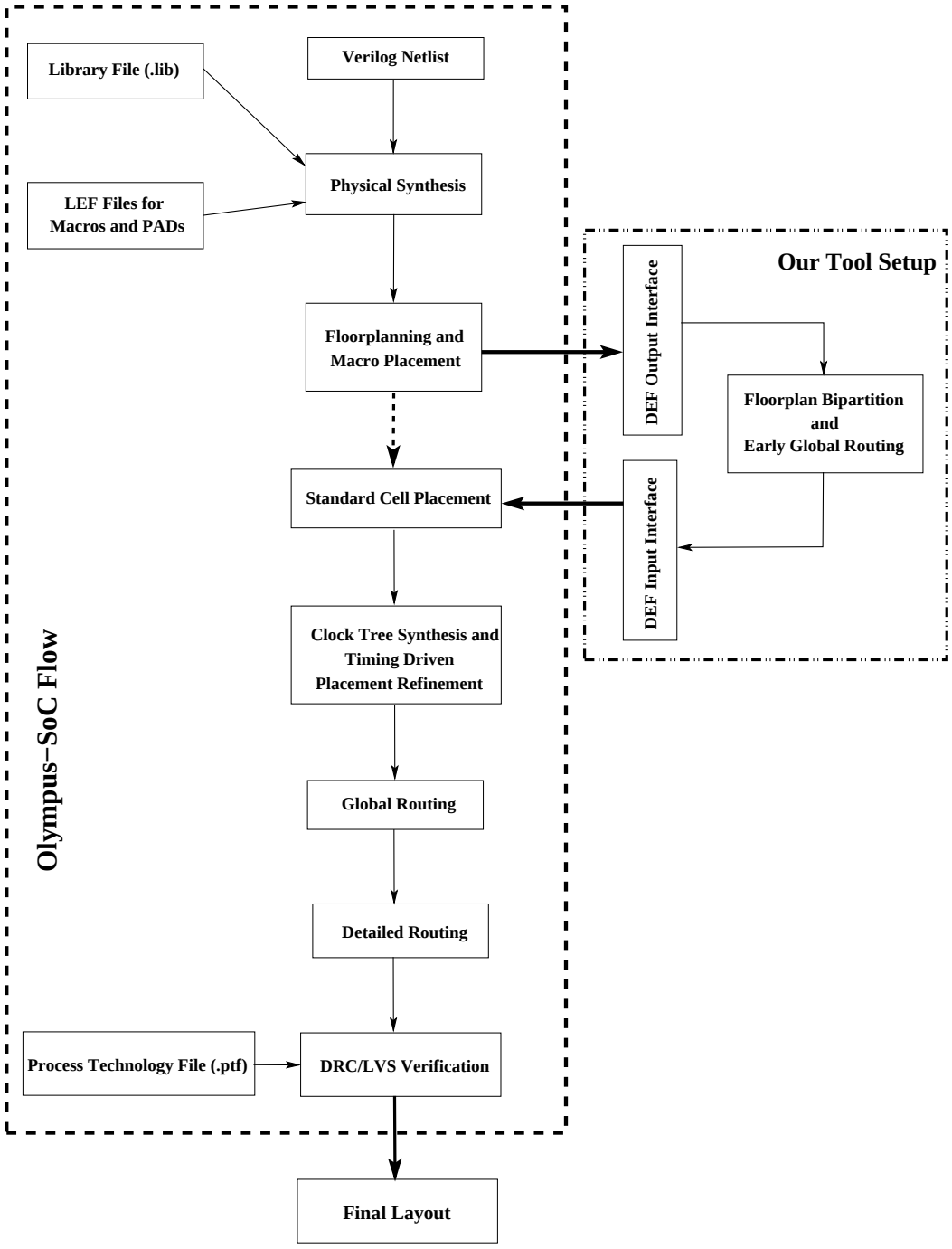


Figure 7.1: Interfacing the Early Global Routing Tools (STAIRoute/HGR) with Olympus-SoC flow [13]

for Olympus-SoC tool was set to be 0.6 for the entire chip area of 0.146 mm^2 (with placeable row area of 77.58K μm^2). There is another target set in this experiment known as non-negative WNS (worst negative slack) and TNS (total negative slack),

apart from the congestion minimization, by restricting both horizontal and vertical congestion to 1.0, similar to that in the EGR framework in this thesis. The corresponding results on the completion of the entire flow are presented in the subsequent section of this chapter.

7.3 Experimental Results

In this section, we present the results from Olympus-SoC tool at the end of physical verification sign-off, indicating DRC/LVS violations. In Table 7.1, we present the final results for two different floorplan instances with different clustering of the standard cells, assuring no DRC/LVS errors. These clusters act as the soft blocks and also provides the guidance for the detailed placement of the standard cells within their respective region boundaries. These clustering also guides us to define the nets for the interconnections among the I/O pads, macro pins and the virtual pins of the soft blocks. In this table, we first study the impact of incorporating STAIRoute as an early global routing (EGR) framework on the final outcome of various key parameters. We compare these results with the same set of results when Olympus-SoC is used to generate the layout of the design without considering the proposed EGR tools STAIRoute/HGR.

In the first set of experiments, we ran STAIRoute on two different floorplan instances, namely #1 and #2, from Olympus in order to obtain the corresponding EGR results. For each instance, we feed these results back to the Olympus-SoC flow prior to the detailed placement of the cells as depicted in Figure 7.1. At this point, we consider two different scenarios on the placement, namely with or without *region blockage constraint*. This blockage constraint feature in Olympus-SoC prohibits the standard cells to be placed within the regions designated as their respective clusters, leaving some empty space for routing around the edges within the boundary of these regions. This makes the placement optimization even harder, increasing timing and iterative placement overhead. This is primarily dictated by the placement utilization values in column 3 and 5 for the respective floorplan instances, showing the target of 60% being grossly violated. The impact of this placement solution on the wirelength, via count and even in average congestion (both in X and Y) are seen to have significantly increased as compared to those obtained by the standalone Olympus-SoC tool, or even those without this constraints. Only important observation in this case

Table 7.1: Impact of STAIRoute with floorplan instances #1 and #2 on the final results obtained by Olympus Flow

Parameter(s)	Olympus-SoC Flow [13]		Olympus-SoC + STAIRoute[a]			
	Instance#1	Instance#2	Instance#1		Instance#2	
	Region Blockage					
	Not Applicable		Yes	No	Yes	No
Standard Cells (Macros)	16298 (4)	16360 (4)	16558 (4)	16517 (4)	16689 (4)	16611 (4)
Nets	16415	16474	16672	16520	16803	16725
Bufffer/Inverter	246/6315	225/6527	225/6331	243/6455	223/6652	225/6575
Placed Area (μm^2) (Buffer area) (μm^2)	45524 (9633)	45909 (10104)	46025 (10217)	45937 (10131)	46167 (10373)	45966 (10157)
Utilization (%)	58.67	59.17	69.30	59.21	72.89	59.24
Wirelength (mm)	273.52	273.02	285.76	273.91	293.58	273.52
Via Count (x10^3)	50.44	52.80	55.59	52.34	55.96	53.13
WNS (ns) TNS (ns)	0.000 0.000	0.000 0.000	0.002 0.000	0.000 0.000	0.002 0.000	0.000 0.000
Avg. Cong. (X) (Y)	0.120 0.113	0.123 0.111	0.124 0.116	0.099 0.100	0.125 0.126	0.122 0.112
Worst Cong. (X) (Y)	0.800 1.000	0.826 0.928	0.708 1.000	0.722 1.000	0.739 0.923	0.791 0.916
DRC/LVS Vio.	No	No	No	No	No	No
CPU time (sec)	2014	1979	843	767	821	688

a - bipartitioning done using $\gamma = 0.5$, and $\beta = 0.2$

is that the worst congestion has improved for both the floorplan instances.

When no such region blockage constraints are used, the placement engine in Olympus-SoC considers the results from EGR tools as its guidance for obtaining a suitable placement solution, aimed to meet both timing and placement density (within 60%) after a number of iterations. The results show that wirelength and via count are similar to those obtained by standalone Olympus-SoC. From these results, we see that this case without any region blockage constraints along the internal boundary of a soft block (region/cluster of standard cells) yields better results than those with the constraint. From the first row, it is noticeable that the standard cell count leading to increased cumulative cell area and hence the number of nets is larger than that in case of Olympus flow; as a result due to the EGR tools than that from standalone Olympus-SoC flow. Despite this difference, the parameter values, for instances with no region blockage constraint, remained competitive, such as utilization % and total buffer/inverter area. If these were same, then the parameters had further scope to improve e.g, better wirelength, via count, and congestion in X/Y directions. Therefore, in the subsequent experiments, we use no blockage constraints for subsequent

placement optimization.

Table 7.2: Impact of STAIRoute and HGR with floorplan instance#1 (without region based placement constraints) on the final results

Parameter(s)	**Olympus-SoC [13]**	**Olympus [13]** + STAIRoute[a]	**Olympus [13]** + HGR[a]
Standard Cells (Macros)	16298 (4)	16517 (4)	16554 (4)
Nets	16415	16520	16749
Buffer/Inverter	246/6315	243/6455	230/6548
Placed Area (Buf area) (μm^2)	45524 (9633)	45937 (10131)	46102 (10299)
Utilization (%)	58.67	59.21	59.42
Wirelength(mm)	273.52	273.91	272.83
Via Count (x10^3)	50.44	52.34	53.01
WNS/TNS (ns)	0.000/0.000	0.000/0.000	0.000/0.000
Avg. Congestion (X/Y)	0.120/0.113	0.099/0.100	0.121/0.113
Worst Congestion (X/Y)	0.800/1.00	0.722/1.000	0.750/0.923
DRC/LVS violations	No	No	No
CPU time (sec)	2014	767	655

[a] - bipartitioning done using $\gamma = 0.5$, and $\beta = 0.2$

We extend these experiments with another early global routing method HGR proposed in Chapter 6 and obtain a similar set of results for HGR running on floorplan instances #1 and #2 respectively for some specific (γ,β) values and compare them with the results obtained by standalone Olympus and STAIRoute+Olympus flows. The corresponding results showing comparison between the flows using HGR+Olympus, STAIRoute+Olympus and standalone Olympus tool are presented in Tables 7.2 and 7.3 for the respective floorplan instances. In this study, we do not impose the region blockage constraint on the placement of standard cells within the regions accommodating the corresponding clusters of standard cells. We also use the same (γ,β) values and the utilization factor.

From these results, we notice very small deviation of wirelength values among all the three flows cited above. However, the via count in both STAIRoute+Olympus and HGR+Olympus are larger than in standalone Olympus for instance #1, while instance #2 shows that via count in standalone Olympus flow is marginally smaller than the other two. Similar results were also obtained in case of average congestion in both X and Y directions, while worst case congestion in X/Y have improved. The final design utilization is also contained within 60% for all the cases. Finally, the runtime in standalone Olympus flow is 3 times more than STAIRoute/HGR based Olympus flow. Moreover, timing was met for all the cases with no negative WNS/TNS values.

Table 7.3: Impact of STAIRoute and HGR with floorplan instance#2 (without region based placement constraints) on final results

Parameter(s)	**Olympus-SoC [13]**	**Olympus [13]** + STAIRoute[a]	**Olympus [13]** + HGR[a]
Standard Cells (Macros)	16360 (4)	16611 (4)	16504 (4)
Nets	16474	16725	16618
Buffer/Inverter	225/6527	225/6575	225/6441
Placed Area (Buf area) (μm^2)	45909 (10104)	45966 (10157)	45834 (10018)
Utilization (%)	59.17	59.24	59.07
Wirelength(mm)	273.02	273.52	273.43
Via Count (x10^3)	52.80	53.13	52.84
WNS/TNS (ns)	0.000/0.000	0.000/0.000	0.004/0.000
Avg. Congestion (X/Y)	0.123/0.111	0.122/0.112	0.122/0.112
Worst Congestion (X/Y)	0.826/0.928	0.791/0.916	0.782/0.909
DRC/LVS violations	No	No	No
CPU time (sec)	1979	688	699

a - bipartitioning done using $\gamma = 0.5$, and $\beta = 0.2$

7.3.1 A Detailed Study

In this section, we study on the key routing metrics for evaluating the implementation of the given verilog design, by running regression on these floorplan instances. In Tables 7.4 and 7.5, we present the final wirelength of the nets obtained Olympus-SoC by using: (i) using early global routing results (STAIRoute/HGR) and (ii) without these results. As cited earlier, these early routing results act as the *guide* to subsequent global and detailed routing steps in Olympus-SoC. These results correspond to the respective floorplan instances #1 and #2 if the design and a set of (γ,β) values used as the floorplan bipartitioning methods (see Chapter 5). We obtain the following results after successful DRC/LVS signoff towards the end of the flow.

In these Tables, we present average, minimum and maximum wirelength computed by Olympus during its optimization steps, both using the results from STAIRoute/HGR and standalone Olympus. Our observation on these results suggest that the minimum wirelength was been obtained immediately after the early global routing results were fed to Olympus and the corresponding routing was done for all the nets connecting the standard cells within the clusters of standard cells. Accordingly, in case of standalone mode, we also observe the minimum wirelength during the initial optimization stages at the initial stage of detailed placement when timing constraints were not imposed and fewer buffers/inverters were used to meet the timing. The final wirelength denotes the overall wirelength of the nets after DRC/LVS signoff by Calibre [83] integrated in this Olympus flow. These results also show that, in all cases, eight metal layers

Table 7.4: Impact of STAIRoute and HGR on wirelength (mm) in floorplan instance#1

Metal	STAIRoute + Olympus			HGR + Olympus			Olympus		
Layer	Final	Min	Max	Final	Min	Max	Final	Min	Max
M_1	2.099	2.000	2.950	2.098	1.999	2.957	3.235	3.110	3.540
M_2	110.098	104.370	115.000	109.969	104.143	115.000	105.522	103.360	107.530
M_3	123.388	115.620	126.890	123.407	115.620	126.897	117.765	111.370	120.560
M_4	21.564	19.080	25.780	21.666	19.080	25.885	25.498	23.380	26.800
M_5	12.928	10.450	20.270	12.937	10.450	20.270	14.025	10.720	16.480
M_6	2.520	2.350	3.710	2.537	2.350	3.762	2.943	2.720	3.530
M_7	0.432	0.390	0.780	0.428	0.383	0.780	0.475	0.130	0.670
M_8	0.169	0.160	0.170	0.167	0.157	0.170	0.150	0.150	0.150
M_9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
M_{10}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Total	273.193	271.370	276.340	273.204	271.370	276.365	269.566	258.850	273.520

were used to route the nets and presents layer wise distribution of the wirelengths. The corresponding minimum, average and maximum wirelength values are similar in standalone mode, and with STAIRoute/HGR based early global routing solutions.

Table 7.5: Impact of STAIRoute and HGR on wirelength (mm) in floorplan instance#2

Metal	STAIRoute + Olympus			HGR + Olympus			Olympus		
Layer	Final	Min	Max	Final	Min	Max	Final	Min	Max
M_1	2.068	1.990	3.050	2.068	1.990	3.050	3.180	3.110	3.180
M_2	106.544	100.970	110.700	106.544	100.970	110.700	105.880	103.520	105.880
M_3	123.921	118.010	127.000	123.921	118.010	127.000	120.560	117.690	120.560
M_4	24.351	22.830	27.360	24.351	22.830	27.360	25.910	26.800	26.800
M_5	13.200	11.180	19.870	13.200	11.180	19.870	14.560	16.480	16.480
M_6	2.997	2.720	4.490	2.997	2.720	4.490	2.720	3.150	2.720
M_7	0.325	0.280	0.590	0.325	0.280	0.590	0.570	0.670	0.570
M_8	0.115	0.110	0.160	0.115	0.110	0.160	0.150	0.150	0.150
M_9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
M_{10}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Total	273.507	271.500	276.720	273.507	271.500	276.720	273.520	271.570	273.520

Similarly, the results on the final, minimum and maximum via count in all three cases are presented in Tables 7.6 and 7.7 for floorplan instances #1 and #2 respectively. Since the results on wirelength show that eight metal layers were used, these tables also ensure that by showing via usage upto V_{78} layer, between M_7 and M_8 routing layers. As per the results, all these vias are single vias, with no double or mult-vias being used to route the nets till M_8. No routing beyond M_8 was done in these examples. This is also evident from the layer wise via distribution.

Table 7.6: Impact of STAIRoute and HGR on via count (x10^3) in floorplan instance#1

Via Layer	STAIRoute + Olympus			HGR + Olympus			Olympus		
	Final	Min	Max	Final	Min	Max	Final	Min	Max
V_{12}	29.274	27.690	29.570	29.264	27.685	29.570	28.200	27.390	28.200
V_{23}	21.062	19.860	21.310	21.062	19.860	21.311	20.120	20.090	20.120
V_{34}	1.317	1.100	1.760	1.320	1.100	1.764	1.520	1.650	1.520
V_{45}	0.437	0.340	0.700	0.437	0.340	0.702	0.500	0.600	0.500
V_{56}	0.067	0.060	0.120	0.067	0.060	0.121	0.080	0.090	0.080
V_{67}	0.011	0.010	0.020	0.011	0.010	0.020	0.020	0.020	0.020
V_{78}	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005
V_{89}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
V_{910}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Total	52.178	49.680	53.010	52.172	49.680	53.010	50.440	49.840	50.440

In these table, we present total as well as layer wise via count, from V_{12} to V_{78} for routing upto M_8. From these results, it is evident that most of the vias (V_{12}, V_{23}) were used, with an average of around 50% and 40% respectively, in connecting the nets running through M_1, M_2 and M_3. The remaining vias are used to make the interconnection of the nets upto M_8. Notably, there are very few vias used as V_{45} and beyond for these interconnections. From these results, we notice that lower layers used slightly more (5%) vias in case of STAIR/HGR than Olympus standalone, while higher layers depict 13% fewer vias. The results on the congestion values are presented

Table 7.7: Impact of STAIRoute and HGR on via count (x10^3) in floorplan instance#2

Via Layer	STAIRoute + Olympus			HGR + Olympus			Olympus		
	Final	Min	Max	Final	Min	Max	Final	Min	Max
V_{12}	29.005	27.610	29.400	29.005	27.610	29.400	27.888	27.110	28.200
V_{23}	21.060	19.970	21.320	21.060	19.970	21.320	19.910	19.340	20.120
V_{34}	1.403	1.250	1.820	1.403	1.250	1.820	1.527	1.410	1.650
V_{45}	0.459	0.380	0.730	0.459	0.380	0.730	0.495	0.390	0.600
V_{56}	0.074	0.060	0.130	0.074	0.060	0.130	0.084	0.080	0.100
V_{67}	0.011	0.010	0.020	0.011	0.010	0.020	0.019	0.010	0.020
V_{78}	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005
V_{89}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
V_{910}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Total	52.010	49.850	52.840	52.010	49.850	52.840	49.918	48.660	50.440

in Tables 7.8 and 7.9. In this table, we capture both average and worst case congestion of the global routing edges, both in X (horizontal) and Y (vertical) directions. These results present the final, minimum and maximum values for all these parameters on all the three cases, namely STAIRoute/HGR-cum-Olympus and standalone Olympus. The tables show that average congestion in both X/Y are similar values for all the

Table 7.8: Impact of STAIRoute and HGR on average and the worst congestion in floorplan instance#1

	EGR + Olympus			HGR + Olympus			Olympus		
Congestion	Final	Min	Max	Final	Min	Max	Final	Min	Max
Avg-X	0.121	0.117	0.122	0.121	0.117	0.122	0.120	0.119	0.120
Avg-Y	0.113	0.113	0.114	0.113	0.113	0.114	0.113	0.112	0.113
Worst-X	0.809	0.750	0.833	0.810	0.750	0.844	0.800	0.750	0.800
Worst-Y	0.989	0.923	1.000	0.986	0.922	1.000	1.000	1.000	1.000

cases. However, in case of instance #1, the final worst congestion in X has improved in case of early global routing using STAIRoute/HGR than in Olympus, while the same in Y increases marginally. Instance #2 exhibit improved worst congestion in Y than in X.

Table 7.9: Impact of STAIRoute and HGR on average and the worst congestion in floorplan instance#2

	EGR + Olympus			HGR + Olympus			Olympus		
Congestion	Final	Min	Max	Final	Min	Max	Final	Min	Max
Avg-X	0.122	0.119	0.123	0.122	0.119	0.123	0.117	0.110	0.120
Avg-Y	0.113	0.112	0.114	0.113	0.112	0.114	0.112	0.110	0.113
Worst-X	0.840	0.750	1.000	0.840	0.750	1.000	0.827	0.750	1.000
Worst-Y	0.903	0.900	0.917	0.903	0.900	0.917	0.977	0.917	1.000

In these experiments, *larger effort* was given in meeting the timing constraint while congestion minimization was given *medium effort* for both the floorplan instances of the design. No negative values were obtained for the timing parameters WNS/TNS and the values fell in the range of $[0.000, 0.004]$ nanoseconds. No violations of DRC/LVS found during physical verification signoff. The average runtime for Olympus, with early global routing, was 10 minutes and 57 seconds with standard deviation of 1 minute and 28 seconds, with negligible time to read DEF files from STAIRoute/HGR interface. On the other hand, Olympus standalone mode takes 32 minutes and 59 seconds for floorplan instance #1 and 33 minutes and 34 seconds for floorplan instance #2 to complete the design flow. It is important to note that average runtime for STAIRoute/HGR running for different (γ,β) values was 13.96 seconds with a standard deviation of 1.66 seconds and time to exchange the DEF files were around 1/2 seconds. Hence, these values were not counted in the overall runtime of the entire flow in all the cases.

7.4 Chapter Summary

In this chapter, we present a case study on an industrial design by incorporating the proposed early global routing methods STAIRoute and HGR, presented in Chapters 4 and 6 respectively, in Olympus-SoC flow using an industry standard design exchange format commonly known as DEF format. Two such interfaces were developed in order to establish a proper mechanism to exchange the information between Olympus-SoC tool and the proposed early global routing methods presented in this thesis.

In this exercise, the input floorplan and the corresponding netlist required by STAIRoute/HGR are provided through the input DEF interface. These methods then generate the corresponding early global routing results in two stages (i) first generating floorplan bipartitioning solutions for identifying the monotone staircase routing regions based on certain values of the trade-off parameters (γ,β), and (ii) then obtaining the multi-layer routing paths of the floorplan level netlist for a set of routing layers. The routing solutions with 100% routing completion from these methods are then fed through the output DEF interface back to the Olympus-SoC tool for subsequent stages to complete. It is important to note that Olympus with STAIRoute/HGR results in increased net count as a result of increased cell count during the subsequent timing driven placement optimization, than in standalone Olympus mode.

Despite the increased cell/net count overhead, the results obtained for two different floorplan instances of the design show no timing violations, while yielding reduced worst congestion and similar average congestion in X/Y direction at the cost of marginally higher netlength and via counts. Moreover, the design placement utilization (placement density) was always constrained within 60% as targeted. We also notice that same number of metal layers used to route the nets, but Olympus with STAIRoute/HGR show fewer nets and hence fewer vias in the higher routing layers. These results were accompanied by no DRC/LVS violations and timing budget were also met with non-negative WNS/TNS values at the end of the flow. The most important factor is that the overall runtime required by the entire flow with early global routing solution is 3x smaller than that for the standalone Olympus flow.

Chapter 8

Conclusions and Future Work

In this thesis, we present a new framework for early global routing (EGR) in a floorplanned design. Alike the existing global routing framework, the proposed EGR framework consists of two stages: (i) identify the routing regions in a given floorplan by a recursive floorplan bipartitioning methods, and (ii) construct a new routing graph that enables this framework to identify the routing paths of a set of nets, abstracted at the floorplanning level of the design, over a predefined set of routing layers. Contrary to the post-placement global routing approaches, this EGR framework gives a new outlook to the existing physical design (PD) flow by facilitating early routability assessment on a given floorplan instance of a design. The corresponding result gives an idea about the quality of the floorplan in terms of the routing metrics. This framework also indicates whether the given floorplan is conducive of producing an acceptable routing solution when the subsequent placement of standard cells is done, considering the early global routing result as a *guidance*. Exploring the scope of DFM awareness during this EGR stage with the help of suitable early abstraction of a few DFM issues may potentially help, in analyzing the challenges associated with the VDSM fabrication processes.

8.1 Contributions of the Thesis

Summary of the contributions made in this thesis are enlisted below:

Faster recursive floorplan bipartitioning framework is presented for defining a set of monotone staircase routing regions in a floorplan, by:

- using breadth first traversal (BFS) or depth first traversal (DFS) on the floorplan topology graph (called block adjacency graph),
- exploring both area and number balanced modes yield better bipartitioning results in terms of *Gain* values as compared to the existing maxflow based approach,
- achieving smaller runtime overhead for DFS over BFS based method, yet both are much faster than the maxflow based approach, and
- obtaining improved *Gain* values for BFS based approach over DFS based approach, for most of the γ values used in this multi-objective optimization problem

The Early Global Routing framework is proposed for early routability assessment on any floorplan instance of a design, irrespective of its sliceability, called STAIRoute, in which we:

- constructed a new routing model from the floorplan bipartitioning results obtained by the proposed methods presented in the thesis, called junction graph, which is again augmented for each net called global staircase routing graph (GRSG),
- computed the capacity of the routing regions from the net cut information,
- proposed a new congestion model that restricts the routing demand in any routing region in any permissible metal layer to maximum routing capacity in that layer,
- obtained the shortest routing paths connecting the pins of each net is identified across multiple metal layers, depending on the prevailing congestion scenario in the routing regions, thereby addressing the pin-access problem at this scope of global routing
- a new multi-terminal net decomposition framework proposed called Staircase Minimal Steiner Topology (SMST),
- incorporated the scope of having different profiles of varying routing capacity across the routing layers and also explored the directional shortest path search by swapping the definition of source and sink vertices during shortest path search in GSRG, and

- obtained results for both reserved and unreserved layer model show encouraging results, ensuring 100% routing completion, no over-congestion, netlength within a fixed factor of the Steiner length computed by a popular method, in addition to significant variation in via count and worst case congestion for different capacity profiles and search directions and using the results from different bipartitioning modes (BFS, DFS)

Early Via minimization is attempted during the proposed early global routing by a new floorplan bipartitioning framework in order to obtain minimal bend monotone staircase pattern routing, and can be seen as an early version of unconstrained via minimization (UVM) problem. This framework:

- defined a new objective of bend minimization in the existing multi-objective floorplan bipartitioning problem, along with the additional trade-off parameter β for this objective,
- presented BFS/DFS based greedy recursive approaches in order to identify a set of optimal monotone staircase regions in the floorplan, for helping STAIRoute to identify minimal bend routing paths for fewer via counts,
- proposed a new staircase wavefront propagation based bipartitioning approach for identifying minimal bend routing regions in a floorplan, by new randomized neighbor search technique,
- obtained encouraging results on each of the objectives as well as a modified version of *Gain* function, for a specified (γ, β) pair, specially for the randomized mode with increased solution space, and
- showed variation of early via count obtained by STAIRoute for different floorplan instances show the effectiveness of these bipartitioning approaches, depending on the values of (γ, β) pairs

DFM aware early routability assessment of a floorplanned design is addressed by:

- proposing a new congestion model for *uniform wire distribution* during early global routing intended for potentially minimizing the surface irregularities due to CMP process, by incorporating different routing costs for terminating and non-terminating nets, proposing greedy pin assignment at

the block boundary etc. in an algorithmic framework WDGRoute similar to STAIRoute

- proposing an extended version of STAIRoute in order to facilitate *over-the-block early global routing* of the nets in a floorplan, by suitably adopting the existing grid graph model based on the floorplan bipartitioning information,
- presenting a hybrid routing model, called HGR, by suitably adopting the proposed junction graph based early global routing for routing lower layers with heavy congestion and routing blockages while the adopted grid graph model used for routing in upper routing layers with minimal/no routing blockages. This method used a similar congestion model of STAIRoute for over-the-block routing for no over congestion and minimal wirelength/via overhead,
- incorporating an early abstraction model of *edge placement error* (EPE) in the congestion model, based on the existing simulation results,
- obtained encouraging results for both CMP and EPE aware routing models on a larger set of floorplanning benchmarks in order to study the impact of the respective DFM effects in early global routing, and
- conducted a comparative study between the proposed EGR methods and some of the popular post-placement academic global routers based on normalized wirelength on a set of relevant floorplanning and placement benchmarks respectively, although this is not a direct comparison between these two global routing frameworks due to different scopes in the existing PD flow.

A Case study on an industrial design is conducted for assessing the impact of the proposed EGR methods (STAIRoute/HGR) by integrating them in the existing PD flow, while:

- using a design implemented in $45nm$ process using an industrial PD tool, called Olympus-SoC and physical verification conducted by the integrated Olympus-Calibre flow,
- developing a new LEF/DEF interface for exchanging information between Olympus and our methods,

- using two different floorplan instances of the design obtained by a predefined settings in Olympus-SoC tool, and design utilization target of 60%,
- showing that the results show significant impact of STAIRoute/HGR on the final metrics such as worst case congestion, average congestion and runtime, over the standalone mode of Olympus-SoC,
- assessing that the wirelength and via count results show slight degradation of wirelength and via count, pertaining to the results obtained by the corresponding timing driven placement stage using EGR as guidance and inserted more cells and hence additional nets than in standalone mode,
- noticing no violations during physical verification stage can be seen, neither any timing violations reported. In some cases, HGR+Olympus achieved positive slack while standalone mode obtained zero slack, and
- finding that the runtime in case of Olympus integrated with EGR is significantly smaller than that of standalone Olympus.

8.2 Future Research Directions

So far, we attempted to incorporate the proposed early global routing (EGR) framework in an industrial PD flow presented in Chapter 7. Our experience with this case study and the existing research works on various optimization approaches in the existing PD flow lead to the following works that we plan to undertake in the near future:

- Integrated floorplan optimization tool with the proposed EGR framework, instead of traditional HPWL based floorplan optimization:
  - In floorplan optimization, HPWL is used as the sole objective. We intend to extend the list of objectives as routed wirelength, via count and even congestion by suitably integrating EGR framework in the floorplan optimization process for better solutions
  - Aim at both block level pin planning for soft blocks and top level pin planning (IO pad placement)
- Early global routing (EGR) aware detailed placement, global and detailed routing framework:

  - Guiding the existing global routing tools for routing the nets connecting to intra-cluster (within the soft blocks) standard cells; also explore the scope of the proposed hybrid routing model for varying number of maximum internal layers used for internal routing
  - Addressing/Exploring pin access problem for intra-cluster nets (connecting the nets starting from a virtual pin at the soft block boundary and terminating on the standard cells within it)
  - Realizing interleaved post-placement global/detailed routing of the nets guided by the early global routing solution
  - Optimizing Buffer placement on longer nets using this framework

- Early abstraction of DFM cost in EGR routing model:
  - Assess the results obtained for early distribution of uniform wire density on the post layout minimal surface irregularities and metal thickness variation due to CMP and the projected dummy fills
  - Enhanced simulation based early modeling of OPC/EPE cost in EGR routing model

- Early global routing (EGR) in three dimensional ($3D$) IC design flow:
  - Realize floorplan bipartitioning and early global routing framework for multi-die 3D ICs
  - Estimate/Minimize the number of Through Silicon Vias (TSV)

References

[1] "Integrated Circuit (IC) Design Flow, wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Physical_design_(electronics)

[2] H. Y. Chen and Y. W. Chang, "Routing for manufacturability and reliability," *IEEE Circuits and Systems Magazine*, vol. 9, no. 3, pp. 20–31, Third 2009.

[3] "Confronting Manufacturing Closure at 32nm and Below." [Online]. Available: http://www.eejournal.com/archives/articles/20091117-mentor

[4] T. C. Chen, G.-W. Liao, and Y. W. Chang, "Predictive Formulae for OPC With Applications to Lithography-Friendly Routing," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 29, no. 1, pp. 40–50, Jan 2010.

[5] "Immersion Lithography, IBM Research." [Online]. Available: http://www.almaden.ibm.com/st/chemistry/lithography/immersion

[6] "Chemical Mechanical Polishing, Wiki-University of Michigan." [Online]. Available: http://lnf-wiki.eecs.umich.edu/wiki/Polishing

[7] "importance of CMP Process." [Online]. Available: http://www.vlsi-expert.com/2015/07/importance-of-cmp-process.html

[8] S. Majumder, S. Sur-Kolay, B. B. Bhattacharya, and S. K. Das, "Hierarchical partitioning of VLSI floorplans by staircases," *ACM Trans. Design Autom. Electr. Syst.*, vol. 12, no. 1, 2007. [Online]. Available: http://doi.acm.org/10.1145/1217088.1217095

[9] M. Hanan, "On Steiners Problem with Rectilinear Distance," *SIAM Journal on Applied Mathematics*, vol. 14, no. 2, pp. 255–265, 1966. [Online]. Available: http://dx.doi.org/10.1137/0114025

[10] C. Chu and Y. C. Wong, "FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 1, pp. 70–83, Jan 2008.

[11] "EE Times: Layer-aware Optimization." [Online]. Available: https://www.eetimes.com/document.asp?doc_id=1279842

[12] J. Mitra, P. Yu, and D. Pan, "RADAR: RET-aware detailed routing using fast lithography simulations," in *Design Automation Conference, 2005. Proceedings. 42nd*, June 2005, pp. 369–372.

[13] "Olympus-SoC tool, Mentor Graphics Inc." [Online]. Available: https://www.mentor.com/products/ic_nanometer_design/place-route/olympus-soc

[14] "Parquet floorplanner and mcnc/gsrc floorplanning benchmarks." [Online]. Available: https://vlsicad.eecs.umich.edu/BK/parquet

[15] Y. Wei, C. Sze, N. Viswanathan, Z. Li, C. Alpert, L. Reddy, A. Huber, G. Tellez, D. Keller, and S. Sapatnekar, "GLARE: Global and local wiring aware routability evaluation," in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, June 2012, pp. 768–773.

[16] "IBM-HB Floorplanning Benchmarks." [Online]. Available: https://cadlab.cs.ucla.edu/cpmo/HBsuite.html

[17] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation*, 2nd ed. Norwell, MA, USA: Kluwer Academic Publishers, 1995.

[18] B. J. Lin, *Optical Lithography, Here Is Why.* Bellingham, WA, USA: SPIE Press, 2010.

[19] Z. Cao, T. T. Jing, J. Xiong, Y. Hu, Z. Feng, L. He, and X. L. Hong, "Fashion: A Fast and Accurate Solution to Global Routing Problem," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 4, pp. 726–737, April 2008.

[20] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Pattern routing: use and theory for increasing predictability and avoiding coupling," *Computer-Aided*

Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 21, no. 7, pp. 777–790, Jul 2002.

[21] M. Pan and C. Chu, "FastRoute: A Step to Integrate Global Routing into Placement," in *Computer-Aided Design, 2006. ICCAD '06. IEEE/ACM International Conference on*, Nov 2006, pp. 464–471.

[22] J. Roy and I. Markov, "High-Performance Routing at the Nanometer Scale," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 6, pp. 1066–1077, June 2008.

[23] K. R. Dai, W. H. Liu, and Y. L. Li, "NCTU-GR: Efficient Simulated Evolution-Based Rerouting and Congestion-Relaxed Layer Assignment on 3-D Global Routing," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 20, no. 3, pp. 459–472, March 2012.

[24] J. Hu, J. A. Roy, and I. L. Markov, "Completing High-quality Global Routes," in *Proceedings of the 19th International Symposium on Physical Design*, ser. ISPD '10. New York, NY, USA: ACM, 2010, pp. 35–41. [Online]. Available: http://doi.acm.org/10.1145/1735023.1735035

[25] M. Cho and D. Pan, "BoxRouter: A New Global Router Based on Box Expansion and Progressive ILP," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 12, pp. 2130–2143, Dec 2007.

[26] M. Cho, K. Lu, K. Yuan, and D. Z. Pan, "BoxRouter 2.0: A Hybrid and Robust Global Router with Layer Assignment for Routability," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 14, no. 2, pp. 32:1–32:21, Apr. 2009. [Online]. Available: http://doi.acm.org/10.1145/1497561.1497575

[27] Y. Xu, Y. Zhang, and C. Chu, "FastRoute 4.0: Global router with efficient via minimization," in *Design Automation Conference, 2009. ASP-DAC 2009. Asia and South Pacific*, Jan 2009, pp. 576–581.

[28] M. D. Moffitt, "MaizeRouter: Engineering an Effective Global Router," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 11, pp. 2017–2026, Nov 2008.

[29] M. M. Ozdal and M. D. F. Wong, "Archer: A History-Based Global Routing Algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 4, pp. 528–540, April 2009.

[30] J. Lu and C. W. Sham, "LMgr: A low-Memory global router with dynamic topology update and bending-aware optimum path search," in *Quality Electronic Design (ISQED), 2013 14th International Symposium on*, March 2013, pp. 231–238.

[31] H. Y. Chen, C. H. Hsu, and Y. W. Chang, "High-performance global routing with fast overflow reduction," in *Design Automation Conference, 2009. ASP-DAC 2009. Asia and South Pacific*, Jan 2009, pp. 582–587.

[32] W. H. Liu, W. C. Kao, Y. L. Li, and K. Y. Chao, "NCTU-GR 2.0: Multithreaded Collision-Aware Global Routing With Bounded-Length Maze Routing," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 32, no. 5, pp. 709–722, May 2013.

[33] T. H. Lee and T. C. Wang, "Congestion-Constrained Layer Assignment for Via Minimization in Global Routing," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 9, pp. 1643–1656, Sept 2008.

[34] C. P. Hsu, "Minimum-Via Topological Routing," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 2, no. 4, pp. 235–246, October 1983.

[35] W. H. Liu, C. K. Koh, and Y. L. Li, "Optimization of placement solutions for routability," in *Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEE*, May 2013, pp. 1–9.

[36] C. C. Chang, J. Cong, Z. Pan, and X. Yuan, "Multilevel global placement with congestion control," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 22, no. 4, pp. 395–409, Apr 2003.

[37] N. Viswanathan, M. Pan, and C. Chu, "FastPlace 3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control," in *Design Automation Conference, 2007. ASP-DAC '07. Asia and South Pacific*, Jan 2007, pp. 135–140.

[38] N. Viswanathan and C. Chu, "FastPlace: efficient analytical placement using cell shifting, iterative local refinement,and a hybrid net model," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 24, no. 5, pp. 722–733, May 2005.

[39] N. Viswanathan, M. Pan, and C. Chu, "FastPlace 2.0: an efficient analytical placer for mixed-mode designs," in *Design Automation, 2006. Asia and South Pacific Conference on*, Jan 2006, pp. 6 pp.–.

[40] M. Pan and C. Chu, "FastRoute 2.0: A High-quality and Efficient Global Router," in *Design Automation Conference, 2007. ASP-DAC '07. Asia and South Pacific*, Jan 2007, pp. 250–255.

[41] Y. Zhang, Y. Xu, and C. Chu, "FastRoute3.0: A fast and high quality global router based on virtual capacity," in *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, Nov 2008, pp. 344–349.

[42] M. Pan and C. Chu, "IPR: An Integrated Placement and Routing Algorithm," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, June 2007, pp. 59–62.

[43] T. Lin and C. Chu, "POLAR 2.0: An effective routability-driven placer," in *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, June 2014, pp. 1–6.

[44] X. He, T. Huang, W. K. Chow, J. Kuang, K. C. Lam, W. Cai, and E. F. Y. Young, "Ripple 2.0: High quality routability-driven placement via global router integration," in *Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEE*, May 2013, pp. 1–6.

[45] Y. Zhang and C. Chu, "GDRouter: Interleaved global routing and detailed routing for ultimate routability," in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, June 2012, pp. 597–602.

[46] J. Cong, J. Fang, M. Xie, and Y. Zhang, "MARS- a multilevel full-chip gridless routing system," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 24, no. 3, pp. 382–394, March 2005.

[47] J. Cong, J. Fang, and Y. Zhang, "Multilevel Approach to Full-chip Gridless Routing," in *Proceedings of the 2001 IEEE/ACM International Conference on Computer-aided Design*, ser. ICCAD '01. Piscataway, NJ, USA: IEEE Press, 2001, pp. 396–403. [Online]. Available: http://dl.acm.org/citation.cfm?id=603095.603178

[48] Y. W. Chang and S. P. Lin, "MR: a new framework for multilevel full-chip routing," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 23, no. 5, pp. 793–800, May 2004.

[49] Y. Xu and C. Chu, "MGR: Multi-level global router," in *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, Nov 2011, pp. 250–255.

[50] J. Cong, J. Fang, and K. Y. Khoo, "DUNE-a multilayer gridless routing system," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 5, pp. 633–647, May 2001.

[51] T. C. Chen and Y. W. Chang, "Multilevel Full-Chip Gridless Routing With Applications to Optical-Proximity Correction," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 6, pp. 1041–1053, June 2007.

[52] H. Y. Chen, S. J. Chou, S. L. Wang, and Y. W. Chang, "A Novel Wire-Density-Driven Full-Chip Routing System for CMP Variation Control," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 2, pp. 193–206, Feb 2009.

[53] K. S. M. Li, Y. W. Chang, C. L. Lee, C. Su, and J. E. Chen, "Multilevel Full-Chip Routing With Testability and Yield Enhancement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 9, pp. 1625–1636, Sept 2007.

[54] J. Hu, M. C. Kim, and I. Markov, "Taming the complexity of coordinated place and route," in *Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEE*, May 2013, pp. 1–7.

[55] C. Lee, "An Algorithm for Path Connections and Its Applications," *Electronic Computers, IRE Transactions on*, vol. EC-10, no. 3, pp. 346–365, Sept 1961.

[56] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, 1st ed. Springer Publishing Company, Incorporated, 2011.

[57] S. S. Sapatnekar, P. Saxena, and R. S. Shelar, *Routing Congestion in VLSI Circuits: Estimation and Optimization.* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[58] Z. Li, C. Alpert, S. Quay, S. Sapatnekar, and W. Shi, "Probabilistic Congestion Prediction with Partial Blockages," in *Quality Electronic Design, 2007. ISQED '07. 8th International Symposium on*, March 2007, pp. 841–846.

[59] J. Westra, C. Bartels, and P. Groeneveld, "Probabilistic Congestion Prediction," in *Proceedings of the 2004 International Symposium on Physical Design*, ser. ISPD '04. New York, NY, USA: ACM, 2004, pp. 204–209. [Online]. Available: http://doi.acm.org/10.1145/981066.981110

[60] J. Westra and P. Groeneveld, "Is probabilistic congestion estimation worthwhile?" in *Proceedings of the 2005 International Workshop on System Level Interconnect Prediction*, ser. SLIP '05. New York, NY, USA: ACM, 2005, pp. 99–106. [Online]. Available: http://doi.acm.org/10.1145/1053355.1053377

[61] M. Marek-Sadowska, "An Unconstrained Topological Via Minimization Problem for Two-Layer Routing," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 3, no. 3, pp. 184–190, July 1984.

[62] "Steiner Tree Problem, wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Steiner_tree_problem

[63] F. K. Hwang, "On Steiner Minimal Trees with Rectilinear Distance," *SIAM Journal on Applied Mathematics*, vol. 30, no. 1, pp. 104–114, 1976. [Online]. Available: http://dx.doi.org/10.1137/0130013

[64] C. Alpert, Z. Li, C. Sze, and Y. Wei, "Consideration of local routing and pin access during VLSI global routing," Apr. 9 2013, US Patent 8,418,113. [Online]. Available: http://www.google.ch/patents/US8418113

[65] T. Taghavi, Z. Li, A. C., G. J. Nam, A. Huber, and S. Ramji, "New placement prediction and mitigation techniques for local routing congestion," in

2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Nov 2010, pp. 621–624.

[66] D. Pandey and S. Peyer, "A novel pin access strategy to improve routability." [Online]. Available: http://priorart.ip.com/IPCOM/000244691

[67] P. Gupta and A. Kahng, "Manufacturing-aware physical design," in *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, Nov 2003, pp. 681–687.

[68] S. H. Teh, C. H. Heng, and A. Tay, "Performance-Based Optical Proximity Correction Methodology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 1, pp. 51–64, Jan 2010.

[69] Y. Qu, S. H. Teh, C. H. Heng, A. Tay, and T. H. Lee, "Timing performance oriented optical proximity correction for mask cost reduction," in *2010 IEEE/SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, July 2010, pp. 99–103.

[70] X. Dong and L. Zhang, "Process-Variation-Aware Rule-Based Optical Proximity Correction for Analog Layout Migration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, no. 99, pp. 1–1, 2016.

[71] A. Hamouda, M. Anis, and K. S. Karim, "Model-Based Initial Bias (MIB): Toward a Single-Iteration Optical Proximity Correction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 10, pp. 1630–1639, Oct 2016.

[72] "Phase-shift Mask, wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Phase-shift_mask

[73] "Phase-shift Mask, SPIE." [Online]. Available: https://spie.org/publications/fg06_p78-80_phase-shift_masks

[74] "OPC/Phase Shift." [Online]. Available: https://www.compugraphics-photomasks.com/our-capabilities/opcphase-shift

[75] "Off-axis illumination, wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Off-axis_illumination

[76] "Immersion Lithography, wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Immersion_lithography

[77] "Immersion Lithography, IBM." [Online]. Available: http://researcher.watson.ibm.com/researcher/view_group_subpage.php?id=3607

[78] "Immersion Lithography, SPIE." [Online]. Available: http://spie.org/publications/fg06_p31-32_immlithography

[79] "Electron Beam Lithography, wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Electron-beam_lithography

[80] K. Yuan, J. S. Yang, and D. Z. Pan, "Double Patterning Layout Decomposition for Simultaneous Conflict and Stitch Minimization," in *Proceedings of the 2009 International Symposium on Physical Design*, ser. ISPD '09. New York, NY, USA: ACM, 2009, pp. 107–114. [Online]. Available: http://doi.acm.org/10.1145/1514932.1514958

[81] Y. Xu and C. Chu, "A Matching Based Decomposer for Double Patterning Lithography," in *Proceedings of the 19th International Symposium on Physical Design*, ser. ISPD '10. New York, NY, USA: ACM, 2010, pp. 121–126. [Online]. Available: http://doi.acm.org/10.1145/1735023.1735054

[82] X. Xu, B. Yu, J. R. Gao, C. L. Hsu, and D. Z. Pan, "PARR: Pin Access Planning and Regular Routing for Self-aligned Double Patterning," in *Proceedings of the 52Nd Annual Design Automation Conference*, ser. DAC '15. New York, NY, USA: ACM, 2015, pp. 28:1–28:6. [Online]. Available: http://doi.acm.org/10.1145/2744769.2744890

[83] "Calibre Tool Suite, Mentor Graphics Inc." [Online]. Available: https://www.mentor.com/products/ic_nanometer_design/verification-signoff

[84] "Calibre InRoute, Mentor Graphics Inc." [Online]. Available: https://www.mentor.com/products/ic_nanometer_design/place-route/calibre-inroute

[85] L. D. Huang and M. Wong, "Optical proximity correction (OPC)-friendly maze routing," in *Design Automation Conference, 2004. Proceedings. 41st*, July 2004, pp. 186–191.

[86] Y. R. Wu, M. C. Tsai, and T. C. Wang, "Maze routing with OPC consideration," in *Design Automation Conference, 2005. Proceedings of the ASP-DAC 2005. Asia and South Pacific*, vol. 1, Jan 2005, pp. 198–203 Vol. 1.

[87] Y. S. Tong and S. J. Chen, "An Automatic Optical Simulation-Based Lithography Hotspot Fix Flow for Post-Route Optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 5, pp. 671–684, May 2010.

[88] M. Cho, K. Yuan, Y. Ban, and D. Pan, "ELIAD: Efficient Lithography Aware Detailed Routing Algorithm With Compact and Macro Post-OPC Printability Prediction," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 7, pp. 1006–1016, July 2009.

[89] D. Ding, J. R. Gao, K. Yuan, and D. Pan, "AENEID: A generic lithography-friendly detailed router based on post-RET data learning and hotspot detection," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, June 2011, pp. 795–800.

[90] T. H. Park, "Characterization and modeling of pattern dependencies in copper interconnects for integrated circuits," Ph.D. dissertation, Dept. EECS, MIT, Cambridge MA, may 2002. [Online]. Available: http://dspace.mit.edu/handle/1721.1/8082

[91] S. Y. Fang, C. W. Lin, G. W. Liao, and Y. W. Chang, "Simultaneous OPC- and CMP-aware Routing Based on Accurate Closed-form Modeling," in *Proceedings of the 2013 ACM International Symposium on International Symposium on Physical Design*, ser. ISPD '13. New York, NY, USA: ACM, 2013, pp. 77–84. [Online]. Available: http://doi.acm.org/10.1145/2451916.2451938

[92] Y. Shen, Q. Zhou, Y. Cai, and X. Hong, "ECP- and CMP-Aware Detailed Routing Algorithm for DFM," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, no. 1, pp. 153–157, Jan 2010.

[93] K. S. Leung, "SPIDER: simultaneous post-layout IR-drop and metal density enhancement with redundant fill," in *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, Nov 2005, pp. 33–38.

[94] C. Dong, Q. Zhou, Y. Cai, and X. Hong, "Wire density driven top-down global placement for CMP variation control," in *Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on*, Nov 2008, pp. 1676–1679.

[95] T. C. Chen, M. Cho, D. Z. Pan, and Y. W. Chang, "Metal-Density-Driven Placement for CMP Variation and Routability," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 12, pp. 2145–2155, Dec 2008.

[96] M. Cho, D. Z. Pan, H. Xiang, and R. Puri, "Wire Density Driven Global Routing for CMP Variation and Timing," in *Computer-Aided Design, 2006. ICCAD '06. IEEE/ACM International Conference on*, Nov 2006, pp. 487–492.

[97] H. Yao, Y. Cai, and X. Hong, "CMP-aware Maze Routing Algorithm for Yield Enhancement," in *VLSI, 2007. ISVLSI '07. IEEE Computer Society Annual Symposium on*, March 2007, pp. 239–244.

[98] C. Feng, H. Zhou, C. Yan, J. Tao, and X. Zeng, "Provably good and practically efficient algorithms for CMP," in *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, July 2009, pp. 539–544.

[99] K. Y. Lee and T. C. Wang, "Post-routing redundant via insertion for yield/reliability improvement," in *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, ser. ASP-DAC '06. Piscataway, NJ, USA: IEEE Press, 2006, pp. 303–308. [Online]. Available: http://dx.doi.org/10.1145/1118299.1118376

[100] K. Y. Lee, C. K. Koh, T. C. Wang, and K. Y. Chao, "Fast and optimal redundant via insertion," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 12, pp. 2197–2208, Dec 2008.

[101] H. A. Chien and T. C. Wang, "Redundant-via-aware eco routing," in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2014, pp. 418–423.

[102] S. Majumder, S. Sur-Kolay, S. C. Nandy, and B. B. Bhattacharya, "On Finding a Staircase Channel with Minimum Crossing

Nets in a VLSI Floorplan," *Journal of Circuits, Systems and Computers*, vol. 13, no. 05, pp. 1019–1038, 2004. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0218126604001854

[103] P. Dasgupta, P. Pan, S. C. Nandy, and B. B. Bhattacharya, "Monotone Bipartitioning Problem in a Planar Point Set with Applications to VLSI," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 7, no. 2, pp. 231–248, Apr. 2002. [Online]. Available: http://doi.acm.org/10.1145/544536.544537

[104] M. Guruswamy and D. F. Wong, "Channel routing order for building-block layout with rectilinear modules," in *IEEE International Conference on Computer-Aided Design (ICCAD-89) Digest of Technical Papers*, Nov 1988, pp. 184–187.

[105] S. Sur-Kolay and B. B. Bhattacharya, "The cycle structure of channel graphs in nonsliceable floorplans and a unified algorithm for feasible routing order," in *IEEE International Conference on Computer Design: VLSI in Computers and Processors*, Oct 1991, pp. 524–527.

[106] S. Majumder, S. Sur-Kolay, B. B. Bhattacharya, and S. C. Nandy, "Area(number)-balanced hierarchy of staircase channels with minimum crossing nets," in *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, vol. 5, 2001, pp. 395–398 vol. 5.

[107] H. Yang and D. F. Wong, "Efficient Network Flow Based Min-cut Balanced Partitioning," in *IEEE/ACM International Conference on Computer-Aided Design*, Nov 1994, pp. 50–55.

[108] H. H. Yang and D. F. Wong, "Balanced partitioning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1533–1540, Dec 1996.

[109] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[110] S. C. Nandy and B. B. Bhattacharya, "On finding an empty staircase polygon of largest area (width) in a planar point-set," *Computational Geometry*, vol. 26, no. 2, pp. 143 – 171, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925772103000154

[111] B. Kar, S. Sur-Kolay, S. H. Rangarajan, and C. Mandal, "A faster hierarchical balanced bipartitioner for VLSI floorplans using Monotone Staircase Cuts," in *Progress in VLSI Design and Test - 16th International Symposium, VDAT 2012, Shibpur, India, July 1-4, 2012. Proceedings*, 2012, pp. 327–336.

[112] B. Kar, S. Sur-Kolay, and C. Mandal, "STAIRoute: Global routing using monotone staircase channels," in *IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2013, Natal, Brazil, August 5-7, 2013*, 2013, pp. 90–95.

[113] S. N. Adya and I. L. Markov, "Fixed-outline floorplanning: Enabling hierarchical design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 6, pp. 1120–1135, Dec 2003.

[114] B. Kar, S. Sur-Kolay, and C. Mandal, "Global Routing Using Monotone Staircases with Minimal Bends," in *2014 27th International Conference on VLSI Design, VLSID 2014, Mumbai, India, January 5-9, 2014*, 2014, pp. 369–374.

[115] "Hasse Diagram, wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Hasse_diagram

[116] R. P. Grimaldi and B. V. Ramana, *Discrete and Combinatorial Mathematics, An Applied Introduction*, 5th ed. New Delhi, India: Pearson Education, 2007.

[117] B. Kar, S. Sur-Kolay, and C. Mandal, "A New Method for Defining Monotone Staircases in VLSI Floorplans," in *2015 IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2015, Montpellier, France, July 8-10, 2015*, 2015, pp. 107–112.

[118] B. Kar, S. Sur-Kolay, and C. Mandal, "An Early Global Routing Framework for Uniform Wire Distribution in SoCs," in *29th International System-On-Chip Conference, SOCC 2016, Seattle, WA, USA, September 6-9, 2016*, 2016, pp. 139–144.

[119] Y. J. Chang, Y. T. Lee, J. R. Gao, P. C. Wu, and T. C. Wang, "NTHU-Route 2.0: A Robust Global Router for Modern Designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 12, pp. 1931–1944, Dec 2010.

[120] S. Batterywala, N. Shenoy, W. Nicholls, and H. Zhou, "Track assignment: a desirable intermediate step between global routing and detailed routing," in *IEEE/ACM International Conference on Computer Aided Design, 2002. ICCAD 2002.*, Nov 2002, pp. 59–66.

[121] Y. Zhang and C. Chu, "RegularRoute: An Efficient Detailed Router Applying Regular Routing Patterns," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 9, pp. 1655–1668, Sept 2013.

[122] B. Kar, S. Sur-Kolay, and C. Mandal, "A Novel EPE Aware Hybrid Global Route Planner after Floorplanning," in *29th International Conference on VLSI Design, VLSID 2016, Kolkata, India, January 4-8, 2016*, 2016, pp. 595–596.

[123] "International Symposium on Physical Design." [Online]. Available: http://www.ispd.cc/

[124] G. J. Nam, C. Sze, and M. Yildiz, "The ispd global routing benchmark suite," in *Proceedings of the 2008 International Symposium on Physical Design*, ser. ISPD '08. New York, NY, USA: ACM, 2008, pp. 156–159. [Online]. Available: http://doi.acm.org/10.1145/1353629.1353663

[125] "NanGate 45*nm* OpenCell Library." [Online]. Available: http://www.nangate.com/?page_id=2325

[126] "LEF/DEF language reference manual (LRM)." [Online]. Available: https://edi.truevue.org/edi/14.17/lefdefref/DEFSyntax.html

Publications from the Thesis

Book Chapters

(BC1) **B. Kar**, S. Sur-Kolay, and C. Mandal, "STAIRoute: Early Global Routing using Monotone Staircases for Congestion Reduction," *to appear as a book chapter titled "Best of ISVLSI 2013" in Springer.*

Refereed Conference Proceedings

(C1) **B. Kar**, S. Sur-Kolay, and C. Mandal, "An Early Global Routing Framework for Uniform Wire Distribution in SoCs," in *29th International System-On-Chip Conference, SOCC 2016, Seattle, WA, USA, September 6-9, 2016*, pp. 139–144.

(C2) **B. Kar**, S. Sur-Kolay, and C. Mandal, "A Novel EPE Aware Hybrid Global Route Planner after Floorplanning," in *29th International Conference on VLSI Design, VLSID 2016, Kolkata, India, January 4-8, 2016*, pp. 595–596.

(C3) **B. Kar**, S. Sur-Kolay, and C. Mandal, "A New Method for Defining Monotone Staircases in VLSI Floorplans," in *2015 IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2015, Montpellier, France, July 8-10, 2015*, pp. 107–112.

(C4) **B. Kar**, S. Sur-Kolay, and C. Mandal, "Global Routing Using Monotone Staircases with Minimal Bends," in *2014 27th International Conference on VLSI Design, VLSID 2014, Mumbai, India, January 5-9, 2014*, pp. 369–374.

(C5) **B. Kar**, S. Sur-Kolay, and C. Mandal, "STAIRoute: Global routing using monotone staircase channels," in *IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2013, Natal, Brazil, August 5-7, 2013*, pp. 90–95.

(C6) **B. Kar**, S. Sur-Kolay, S. H. Rangarajan, and C. Mandal, "A faster hierarchical balanced bipartitioner for VLSI floorplans using Monotone Staircase Cuts," in *Progress in VLSI Design and Test - 16th International Symposium, VDAT 2012, Shibpur, India, July 1-4, 2012. Proceedings*, pp. 327–336.

BIO-DATA

Name:	**Bapi Kar**
Sex:	Male
Date of Birth:	18-03-1980
Father's Name:	Bablu Kar
Nationality:	Indian
Permanent Address:	136G/1, Sarsuna Main Road, P.O. and P.S.- Sarsuna, Kolkata West Bengal, INDIA - 700061
Email:	bapi.kar@gmail.com and bapik@iitkgp.ac.in

Educational Qualifications:

Sl No.	University/Institute	Degree	Year
1	Jadavpur University, Kolkata India	BE (Instrumentation Engineering)	1998-2002

I am pursuing PhD in the field of VLSI Physical Design Automation in Computer Science and Engineering department (erstwhile School of Information Technology) at IIT Kharagpur, India. I completed my bachelor degree (BE) in Instrumentation (and Electronics) Engineering from Jadavpur University, Kolkata, India in 2002 and was awarded with University Gold Medal for standing first in the class. After graduating from Jadavpur University, I worked in a couple of multinational companies, in the field of VLSI design and verification, both in India and abroad. My professional career spanned about 6 years and 4 months.

In early 2009, I decided to pursue further studies at IIT Kharagpur and enrolled in PhD program at School of Information Technology under the supervision of Prof. Chittaranjan Mandal and Prof. Susmita Sur-Kolay of ISI Kolkata in Spring 2009-2010.