UNIVERSITY OF CALIFORNIA, SAN DIEGO

Physical Design and Technology Optimizations for Advanced VLSI Manufacturing

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

in

Electrical Engineering (Computer Engineering)

by

Hye In Lee

Committee in charge:

Professor Andrew B. Kahng, Chair
Professor Chung-Kuan Cheng
Professor Ian A. Galton
Professor Rajesh K. Gupta
Professor Ryan Kastner

2018

The dissertation of Hye In Lee is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2018

DEDICATION

To my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor Professor Andrew B. Kahng for his continuous support and invaluable advice throughout my Ph.D. study. I have learned many lessons from him not only on research but also on life.

I would like to thank my family: my husband Kyu Hyun Kim and our loving son, Han Kim; my father Si Ryong Lee, my mother Cheong Hui Lee and my brother Sang Gyu Lee; my father-in-law Nam Hee Kim and my mother-in-law Ho Sung Kim for their endless support and encouragement.

I would like to thank my fellow labmates in the UCSD VLSI CAD Laboratory (Kwangsoo Han, Minsoo Kim, Uday Mallappa, Vaishnav Srinivas, Lutong Wang, Mingyu Woo, Bangqi Xu and Payal Agarwal) and former lab members (Dr. Tuck-Boon Chan, Dr. Wei-Ting (Jonas) Chan, Dr. Jiajia Li, Dr. Ilgweon Kang, Professor Seokhyeong Kang, Mulong Luo, Dr. Siddhartha Nath, Tushar Shah, Yaping Sun and Sriram Venkatesh) for their assistance and enthusiastic discussions.

My sincere thanks also go to my thesis committee members Professor Chung-Kuan Cheng, Professor Rajesh Gupta, Professor Ian A. Galton and Professor Ryan Kastner for their time, encouragement and insightful comments.

Last, but not least, I would like to thank my industrial collaborators (Peter Debacker, Dr. Hamed Fatemi, Professor Jose Pineda de Gyvez and Dr. Praveen Raghavan) for their invaluable guidance and feedback in many of my research projects.

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

Chapter 2 contains the reprint of Andrew B. Kahng and Hyein Lee, "Minimum Implant Area-Aware Gate Sizing and Placement", *Proc. ACM Great Lakes Symposium on Very Large Scale Integration*, 2014. Chapter 2 also contains the draft submitted to *Integration, the VLSI Journal*, Hamed Fatemi, Andrew B. Kahng, Hyein Lee, Jiajia Li and José Pineda de Gyvez, "Enhancing Sensitivity-Based Power Reduction for an Industry IC Design Context", 2018. Chapter 2 also contains the draft submitted to *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Hamed Fatemi, Andrew B. Kahng, Hyein Lee and José Pineda de Gyvez, "Heuristic Methods for Fine-Grain Exploitation of FDSOI", 2018. The dissertation author is a main contributor to, and a primary author of, each of these papers.

Chapter 3 contains reprints of Peter Debacker, Kwangsoo Han, Andrew B. Kahng, Hyein Lee, Praveen Raghavan and Lutong Wang, "Vertical M1 Routing-Aware Detailed Placement for Congestion and Wirelength Reduction in Sub-10nm Nodes", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2017; and Kwangsoo Han, Andrew B. Kahng and Hyein Lee, "Scalable Detailed Placement Legalization for Complex Sub-14nm Constraints", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2015. The dissertation author is a main contributor to, and a primary author of, each of these papers.

Chapter 4 contains reprints of Alex Kahng, Andrew B. Kahng, Hyein Lee and Jiajia Li, "PROBE: A Placement, ROuting, Back-End-of-line Measurement Utility", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37(7) 2018; Kwangsoo Han, Andrew B. Kahng, Hyein Lee and Lutong Wang, "Performance- and Energy-Aware Optimization of BEOL Interconnect Stack Geometry in Advanced Technology Nodes", *Proc. International Symposium on Quality Electronic Design*, 2017; and Kwangsoo Han, Andrew B. Kahng and Hyein Lee, "Evaluation of BEOL Design Rule Impacts Using an Optimal ILP-Based Detailed Router", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2015. The dissertation author is a main contributor to, and a primary author of, each of these papers.

My coauthors (Mr. Peter Debacker, Dr. Hamed Fatemi, Professor José Pineda de Gyvez, Mr. Kwangsoo Han, Mr. Alex Kahng, Professor Andrew B. Kahng, Dr. Jiajia Li, Dr. Praveen Raghavan and Mr. Lutong Wang, listed in alphabetical order) have all kindly approved the inclusion of the aforementioned publications in my thesis.

VITA

2003	B. S., Electronics Engineering, Yonsei University, Seoul, South Korea
2005	M. S., Electronics Engineering, Korea Advanced Institute of Science & Technology, Daejeon, South Korea
2015	C. Phil., Electrical Engineering (Computer Engineering), University of California, San Diego
2018	Ph. D., Electrical Engineering (Computer Engineering), University of California, San Diego

All papers co-authored with my advisor Professor Andrew B. Kahng have authors listed in alphabetical order.

PUBLICATIONS

A. Kahng, A. B. Kahng, **H. Lee** and J. Li, "PROBE: A Placement, ROuting, Back-End-of-line Measurement Utility", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37(7) (2018), pp. 1459-1472.

C. Han, K. Han, A. B. Kahng, **H. Lee**, L. Wang and B. Xu, "Optimal Multi-Row Detailed Placement for Yield and Model-Hardware Correlation Improvements in Sub-10nm VLSI", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2017, pp. 667-674.

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

P. Debacker, K. Han, A. B. Kahng, **H. Lee**, P. Raghavan and L. Wang, "MILP-Based Optimization of 2D Block Masks for Timing-Aware Dummy Segment Removal in Self-Aligned Multiple Patterning Layouts", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 36(7) (2017), pp. 1075-1088.

K. Han, A. B. Kahng, **H. Lee** and L. Wang, "Performance- and Energy-Aware Optimization of BEOL Interconnect Stack Geometry in Advanced Technology Nodes", *Proc. International Symposium on Quality Electronic Design*, 2017, pp. 104-110. (**Invited Paper**)

A. B. Kahng, **H. Lee** and J. Li, "Measuring Progress and Value of IC Implementation Technology", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2016, pp. 27:1-27:8. (**Invited paper**)

K. Han, A. B. Kahng and **H. Lee**, "Scalable Detailed Placement Legalization for Complex Sub-14nm Constraints", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2015, pp. 867-873.

K. Han, A. B. Kahng, **H. Lee** and L. Wang, "ILP-Based Co-Optimization of Cut-Mask Layout, Dummy Fill and Timing for Sub-14nm BEOL Technology", *Proc. SPIE/BACUS Symposium on Photomask Technology and Management*, 2015, pp. 96350E:1-96350E:14.

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

A. B. Kahng and **H. Lee**, "Minimum Implant Area-Aware Gate Sizing and Placement", *Proc. ACM Great Lakes Symposium on Very Large Scale Integration*, 2014, pp. 57-62.

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

A. B. Kahng and **H. Lee**, "Margin Recovery with Flexible Flip-Flop Timing", *Proc. International Symposium on Quality Electronic Design*, 2014, pp. 496-503.

A. B. Kahng, S. Kang, **H. Lee**, I. L. Markov and P. Thapar, "High-Performance Gate Sizing with a Signoff Timer", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2013, pp. 450-470.

A. B. Kahng, S. Kang and **H. Lee**, "Smart Non-Default Routing for Clock Power Reduction", *Proc. ACM/IEEE Design Automation Conference*, 2013, pp. 91:1-91:7.

A. B. Kahng, S. Kang, **H. Lee**, S. Nath and J. Wadhwani, "Learning-Based Approximation of Interconnect Delay and Slew in Signoff Timing Tools", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2013, pp. 1-8.

K.-T. Do, J. Y. Choi, S. Kim, **H. Lee**, H.-S. Won and K.-M. Choi, "A Practical Framework for Statistical Leakage Estimation", *Proc. ACM/IEEE Design Automation Conference User Track*, 2010.

H. Lee, S. Paik and Y. Shin, "Pulse Width Allocation and Clock Skew Scheduling: Optimizing Sequential Circuits based on Pulsed Latches", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 29(3) (2010), pp. 355-366.

H. Lee, S. Paik and Y. Shin, "Pulse Width Allocation with Clock Skew Scheduling for Optimizing Pulsed Latch-based Sequential Circuits", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 224-229.

ABSTRACT OF THE DISSERTATION

Physical Design and Technology Optimizations for Advanced VLSI Manufacturing

by

Hye In Lee

Doctor of Philosophy in Electrical Engineering (Computer Engineering)

University of California, San Diego, 2018

Professor Andrew B. Kahng, Chair

Recent years have seen a significant slowdown of density scaling in advanced semiconductor integrated-circuit products, despite multiple innovations in patterning technologies, device and cell architectures, and design methodologies. Designers are unable to fully leverage the potential power, performance, area and cost benefits offered by new process technologies. Root causes of this inability include the explosion of scenarios in timing signoff, front-end-of-line (FEOL) layout rules that affect placement, sizing-placement interactions that require new co-optimizations, back-end-of-line (BEOL) layout rules and cell height scaling that impact routing, and the increasingly dominant role of BEOL parasitics on final design quality. To address these challenges for modern system-on-chip physical design and signoff in advanced manufacturing nodes, new design optimization techniques as well as methodologies for design-

technology co-optimization are required. Accordingly, this thesis presents new physical optimization and evaluation methodologies, organized according to three main thrusts.

To address the explosion of corners and modes in timing signoff and the emergence of new sizing-placement interactions, the *post-placement gate sizing optimization* thrust of this thesis presents a gate sizing optimization considering multi-corner multi-mode constraints; a minimum implant rule-aware gate sizing and placement co-optimization; and heuristics for potential fine-grain exploitation of FDSOI technologies.

To address the challenges to scaling brought by new placement rules and reduced-track cell architectures, the *detailed placement optimization* thrust of this thesis presents an integer linear programming-based incremental detailed placement optimization that considers inter-row and intra-row placement constraints; and a detailed placement optimization that reduces wirelength in the context of new cell architectures with vertical M1 pins.

To address the need for design-technology co-optimization, the *evaluation of design enablement* thrust of this thesis presents analyses of impacts of patterning technology choices and associated routing rules on physical implementation density; a study of impacts of BEOL dimensions on block-level power and area; and a methodology for assessment of routing capacity of a BEOL stack as well as inherent capability of routers.

Chapter 1

Introduction

In recent years, the semiconductor industry has struggled to continue the scaling trajectory of Moore's Law, i.e., 2× transistor layout density in each successive technology node. Density scaling in recent technology nodes has been achieved using a variety of levers, including new patterning technologies, new device and cell architectures, and new design methodologies. This has come at the cost of enormous R&D investments in manufacturing technology and design enablement. However, despite these efforts, the past decade has seen a notable slowdown in the scaling of actual IC product *quality of results* (QoR). This slowdown is due to a weakened ability of design methodologies to exploit the "available scaling" afforded by process and device innovation. In other words, current design enablements are unable to extract sufficient power, performance, area and cost (PPAC) benefits from new nodes. As depicted in Figure 1.1, there is a significant gap between the *actual* density scaling and the *available* scaling from technology. Since 2008, the actual density scaling (red squares) has slowed down to 1.6× per node, in contrast to 2× per node of available density scaling (gray arrow).

This thesis addresses root causes of the slowdown in scaling of IC design quality, and proposes improved design methodologies to tackle physical design challenges in advanced technology nodes.

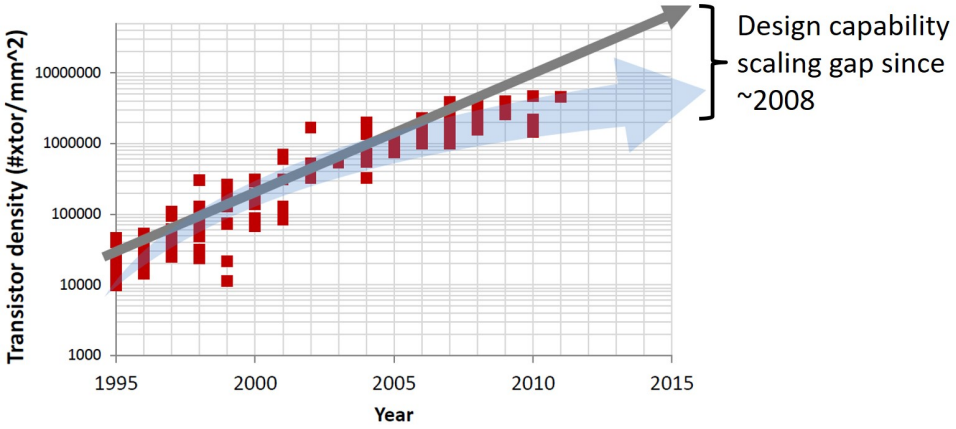


Figure 1.1: Design capability gap [182][81]. The gray arrow shows the "available" density scaling, and the red squares show the "actual" density scaling, in MPU products.

1.1 New Challenges in Advanced Technology Nodes

Root causes of the slowdown in design scaling include the explosion of scenarios in timing signoff, front-end-of-line (FEOL) layout rules that affect placement, sizing-placement interactions that require new co-optimizations, back-end-of-line (BEOL) layout rules and cell height scaling that impact routing, and the increasingly dominant role of BEOL parasitics on final design quality.

1.1.1 Challenges in Gate Sizing Optimization

Discrete gate sizing has been widely adopted for timing and power optimization. The gate sizing optimization has been studied for many years in academia. However, with new timing signoff challenges introduced by the explosion of signoff modes and corners, and with new placement constraints in particular technologies (i.e., Fully Depleted Silicon On Insulator (FDSOI) and sub-22nm), traditional gate sizing methods are no longer suitable for optimization of current real-world designs.

Explosion of Signoff Modes and Corners

Design methodologies for complex modern systems-on-chip (SOCs) introduce a large number of signoff modes and corners, resulting in a huge burden for timing signoff and gate sizing optimization tools. A primary cause of this complexity is the need for extreme low-power consumption in modern SoC designs, such as mobile and IoT products. Multiple scenarios or modes are mandatory, along with mode-specific

design optimizations. For example, mobile products can have sleep and functional modes, and mode-specific power minimization is required to maximize battery life. Modern SoC products can have dozens of modes, each with its own signoff criteria; these modes typically include functional (scenario-based, overdrive, underdrive) and test (scan, at-speed, BIST) modes.

In addition to multiple functional modes, multiple supply voltages and multiple power domains increase voltage corners. In recent technologies, particular with FinFET devices, the use of extreme supply voltage scaling has completely changed the delay-power tradeoff in gate sizing, and has brought many new challenges (e.g., slew times and gate-dominance vs. wire-dominance across signoff corners) to gate sizing optimizations. The growing impact of manufacturing variations in back-end-of-line (BEOL) and front-end-of-line (FEOL) processes further increases the number of corners that must also be considered during design optimization and signoff. For example, BEOL corners now include Cw, Ccw, Cb, RCw, RCb, etc.; FEOL corners now include FF, SS, TT, FS, SF, SSG, FFG, etc.

Gate Sizing and Placement Interaction

Continued technology scaling has been possible only with introduction of multiple-patterning technologies and complex design rules. In older technologies before the $22nm$ node, standard-cell layouts are "composable by construction": arbitrary placements of cells in rows are legal and manufacturable, unless there is an overlap between two cells. However, in advanced technology nodes, the individual cell size has continued to decrease relative to minimum patternable feature sizes. As a result, free composability of cells in placement rows is no longer guaranteed, and placement legality can depend on how a cell is placed in relation to its neighbors. This leads to new challenges for physical design, e.g., interdependencies between gate sizing and detailed placement. Two prominent examples of new placement constraints are the *minimum implant area* (MinIA) constraint and the V_t abutment rule in FDSOI.

Implant (active) layers, which indicate regions for ion implantation, determine the threshold voltage (V_t) of transistors. The implant layer dimension is typically matched to the width of standard cells if the cells consist of single-V_t transistors. Due to the limitation of patterning technologies, minimum width rules for implant layers are introduced to form legal shapes in terms of patterning. Such minimum width rules have become smaller than the minimum cell width in sub-$22nm$ – such that narrow cells

cannot be sandwiched between different-V_t cells. As a result, sizing problem formulations of downsizing and V_t-swapping operations (which by definition do not cause overlaps, and are hence benign in older technologies) must now comprehend spatially adjacent cell instances and whitespace in order to avoid creation of sandwiched narrow cells.

Another example of sizing-placement interaction can be seen in FDSOI. Due to the unique FDSOI device structure that does not have a body node, low-V_t devices are implemented using a special *flip well* structure, while regular V_t devices are implemented using the conventional well. As the wells are flipped, abutting low V_t and regular V_t cells induces a well bias conflict. Due to these constraints, sizing optimizations must have region awareness: low-V_t cells must be spatially contiguous, forming islands in the placement.

1.1.2 Challenges in Placement Optimization

Technology scaling to $10nm$ and below introduces complex intra-row and inter-row constraints in standard-cell detailed placement. Examples of such constraints are found in rules for drain-drain abutment, minimum implant region area and width, oxide diffusion (OD) notching and jogging. These new rules change the definition of "legal" placement, which was simply "non-overlapping" placement in older technologies. Detailed placers must now consider intra-row and inter-row constraints during the optimization to maintain the wirelength / timing / power QoR while achieving legal placements.

Cell architectures also have been changed for better detailed routing. For example, "middle-of-line" layers below M1 are used to gain additional routing resources. New cell architectures wherein inter-row M1 routing is allowed force consideration of vertical alignment of cells. The rapid emergence of new placement rules and new cell architectures motivates the introduction of a new final legalization phase for standard-cell placement tools in advanced (particularly, $10nm$ and $7nm$) foundry nodes.

1.1.3 Challenges in Design-Technology Co-optimization

In advanced technology nodes, BEOL interconnect geometry has become a key lever for design enablement, and a focal point for design-technology co-optimization, due to its significant impact on physical design QoR. Complex design rules and more pervasive use of multi-patterning in the BEOL has

increased the difficulty of maintaining high layout densities. Intuitively, emerging constraints such as unidirectional patterning or increased via spacing will decrease achievable density of the final place-and-route solution, worsening die area and product cost. Also, the rapid increase of interconnect RC leads to not only performance loss from interconnect delay increase, but circuit power and area degradation as well. Optimization of BEOL dimensions (i.e., wire width, spacing and thickness subject to a given layer's pitch constraint) across the entire metal layer stack is crucial to achieve better product performance, power, area and cost.

1.2 This Thesis

To address the new challenges in modern SoC physical design and signoff in advanced nodes, this thesis presents new physical optimization and technology assessment methodologies. Figure 1.2 illustrates the scope and organization of this thesis, in which three main thrusts respectively address three major challenges:

- Post-placement gate sizing optimization;
- Detailed placement optimization; and
- Evaluation of design enablement.

To address the explosion of corners and modes in timing signoff and the emergence of new sizing-placement interactions, the *post-placement gate sizing optimization* thrust of this thesis presents a gate sizing optimization considering multi-corner multi-mode constraints; a minimum implant rule-aware gate sizing and placement co-optimization; and heuristics for potential fine-grain exploitation of FDSOI technologies.

To address the challenges to scaling brought by new placement rules and reduced-track cell architectures, the *detailed placement optimization* thrust of this thesis presents an integer linear programming-based incremental detailed placement optimization that considers inter-row and intra-row placement constraints; and a detailed placement optimization that reduces wirelength in the context of new cell architectures with vertical M1 pins.

To address the need for design-technology co-optimization, the *evaluation of design enablement* thrust of this thesis presents analyses of impacts of patterning technology choices and associated routing rules on physical implementation density; a study of impacts of BEOL dimensions on block-level power and area; and a methodology for assessment of routing capacity of a BEOL stack as well as inherent capability of routers.

The remainder of this thesis is organized as follows.

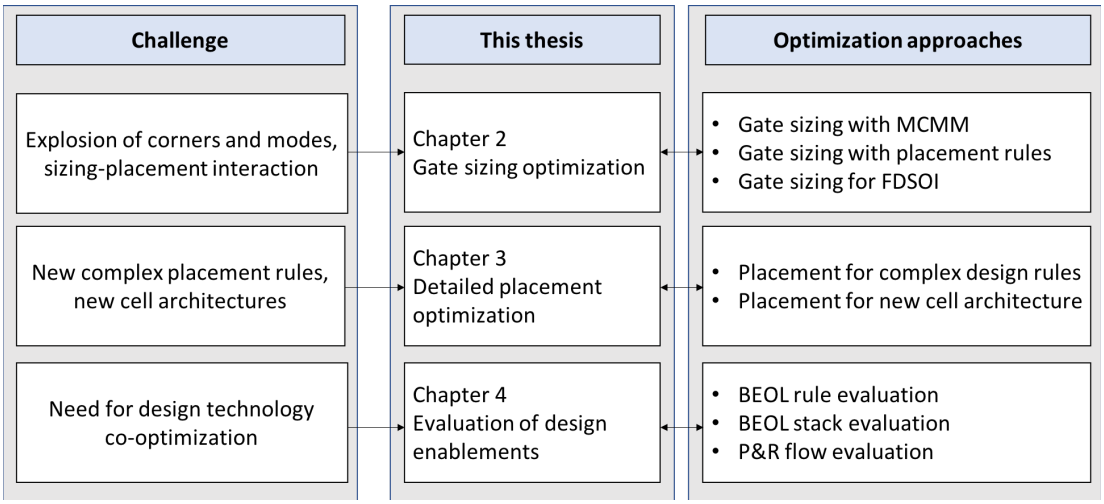


Figure 1.2: Scope and organization of this thesis.

- Chapter 2 presents three distinct gate sizing methodologies that address new challenges in timing signoff, and the interaction between sizing and detailed placement. First, we study important constraints of modern industrial designs that are generally not comprehended by previous academic sizing works. In our study, we point out that various optimization techniques used in academic sizers can fail to offer benefits in product design contexts due to differences in the underlying optimization formulation and constraints. To address this gap, we develop a new robust academic sizer, *Sizer*, from a fresh implementation of *Trident* [84]. Experimental results show that Sizer is able to achieve up to 10% leakage power and 4% total power reductions compared to leading commercial tools on designs implemented with foundry technologies, and 7% leakage power reduction on a modern industrial design in the multi-corner multi-mode (MCMM) context. Second, we propose heuristic methods that fix MinIA violations and reduce power with gate sizing, while minimizing placement perturbations

that potentially create extra timing violations. Compared to recent versions of commercial place-and-route (P&R) tools, our methodologies achieve significant reductions (up to 100%) in the number of MinIA violations while satisfying timing and power constraints. Third, we study heuristic methods for potential exploitation of fine-grained mixed-Vt (and body biasing) in FDSOI implementation, via a novel "speed domain partitioning" (SDP) problem formulation. We explore a broad space of implementation flows, then perform a detailed investigation of two implementation flows: an integer linear programming (ILP)-based approach, and a sensitivity function-based heuristic approach. For implementations using "generic" library options, up to 20% speed improvement with 53% LL region area is seen for one out of four testcases studied. For implementations using "rich" library options, up to 7% speed improvement with 26% LL region area is achieved. In our experiments, we observe that outcomes are strongly library- and design-dependent. We therefore provide a discussion of root-cause, intrinsic difficulties of fine-grained exploitation of mixed-Vt in FDSOI technology. We furthermore suggest a "decision tree" to help assess a design's amenability to fine-grained mixed-Vt (as well as body-biasing based) implementation, and to help guide design flow selection for better design QoR.

- Chapter 3 presents two distinct methodologies for detailed placement optimization for advanced VLSI manufacturing. First, we develop a mixed integer-linear programming (MILP)-based placer, called **DFPlacer**, for final-phase design rule violation (DRV) fixing. DFPlacer finds (near-)DRV-free solutions that consider various complex FEOL layout constraints including minimum implant width, drain-drain abutment, and oxide diffusion jogs. To overcome the runtime limitation of MILP-based approaches, we implement a distributable optimization strategy based on partitioning of the block layout into windows of cells that can be independently legalized. Using layouts in an abstracted $7nm$ library, we find that DFPlacer fixes 99% of DRVs on average with minimal impacts on area and timing. We also study an area-DRV tradeoff between two types of standard-cell library strategies, namely, with and without dummy poly gates. Second, we propose again a MILP-based, detailed placement optimization to maximize direct vertical M1 routing utilization for congestion and wirelength reduction.

- Chapter 4 presents three distinct evaluation methodologies for design flows and technology enablements. First, we study impacts of patterning technology choices and associated design rules on physical implementation density, with respect to *cost-optimal* design rule-correct detailed routing. A key contribution is an integer linear programming (ILP) based optimal router (*OptRouter*) which considers complex design rules that arise in sub-20nm process technologies. Using OptRouter, we assess wirelength and via count impacts of various design rules (implicitly, patterning technology choices) by analyzing optimal routing solutions of clips (i.e., switchbox instances) extracted from post-detailed route layouts in an advanced technology. Second, we study BEOL interconnect stack geometry by exploring wire aspect ratio (AR) and wire line-space duty cycle (DC). We perform SPICE-based analyses of timing path delays to find delay- or power-optimal (AR,DC) combinations, and also perform block-level studies with placed and routed designs. Based on our experimental results, we provide various insights on BEOL stack geometry: (i) optimal (AR,DC) for a given wire pitch with respect to power and delay; (ii) sensitivities of optimal (AR,DC) to circuit parameters (e.g., driver strength, input slew, output load, wirelength); (iii) optimal (AR,DC) when multiple interconnect layers are considered; and (iv) potential impacts of BEOL stack optimizations within future design-aware manufacturing and/or manufacturing-aware design methodologies. Third, we propose a systematic framework to measure routing capacity of a BEOL stack as well as inherent capability of routers. Based on our experimental results, we observe consistent results across mesh-like placement and placements from various placers. Also, our proposed framework enables new insights into important questions regarding BEOL stack options. Using our framework, we further empirically study the relation between the routing hotspot size and routing failure. Lastly, we present an analytical study based on exponentiation of a Markov transition matrix about the impact of design size on routing failure.

- Chapter 5 concludes the thesis.

Chapter 2

Gate Sizing Optimizations for Advanced VLSI Technologies

Discrete gate sizing has been widely adopted for timing and power optimization. The gate sizing optimization has been studies for many years in academia. However, with the new timing signoff challenge that is introduced by the explosion of signoff modes/corners and new placement constraints in particular technologies (i.e., FDSOI and sub-$22nm$), traditional gate sizing work is no longer suitable for current real-world designs. In particular, the new placement constraints induce a new problem formulation wherein gate sizing and detailed placement are tightly linked. Examples of the new placement constraints include the *minimum implant area* (MinIA) constraint and the V_t abutment rule in FDSOI. More specifically, with MinIA rule, a narrow cell cannot be sandwiched between different-V_t cells. In FDSOI, different-V_t (i.e., LL, LR) devices must be isolated from each other, which makes realization of fine-grained mixed-V_t and body-biasing in layout extremely challenging.

This chapter presents three distinct gate sizing methodologies that address new challenges in timing signoff and the interaction between sizing and detailed placement. First, we study important constraints of modern industrial designs that are generally not comprehended by previous academic sizing works. In this study, we point out that various optimization techniques used in academic sizers can fail to offer benefits in product design contexts due to differences in the underlying optimization formulation and constraints. To

address this gap, we develop a new robust academic sizer, *Sizer*, from a fresh implementation of *Trident* [84]. Experimental results show that Sizer is able to achieve up to 10% leakage power and 4% total power reductions compared to leading commercial tools on designs implemented with foundry technologies, and 7% leakage power reduction on a modern industrial design in the multi-corner multi-mode (MCMM) context. Second, we propose heuristic methods that fix MinIA violations and reduce power with gate sizing while minimizing placement perturbation to avoid creating extra timing violations. Compared to recent versions of commercial P&R tools, our methodologies achieve significant reductions (up to 100%) in the number of MinIA violations under timing/power constraints. Third, we study heuristic methods aimed at exploitation of fine-grained mixed-Vt in FDSOI implementation. We propose a novel speed domain partitioning (SDP) problem formulation that comprehends the spatial contiguity restrictions arising from flip-well structure of LL regions in popular 28nm commercial FDSOI offerings. We explore a wide space of implementation flows that include an Integer Linear Programming (ILP)-based approach, and a heuristic (sensitivity-based) optimization. For implementations using generic library options, up to 20% speed improvement with 54% LL region is seen for one out of four testcases studied. For implementations using rich library options, up to 7% speed improvement with 26% LL region is achieved. We further provide a discussion that summarizes rootcause, intrinsic difficulties of fine-grained exploitation of mixed-V_t.

2.1 Enhancing Sensitivity-Based Power Reduction for an Industry IC Design Context

Discrete gate sizing, i.e., change of gate width and length, and V_t type, has been widely adopted for timing and power optimization. Gate sizing optimization can be applied at every design stage, e.g., post-synthesis, post-placement and timing ECO. It is especially suitable for late-stage, post-routing optimization since it incurs relatively small perturbation to the design netlist and layout compared to other optimizations such as logic restructuring and buffer insertion. The importance of gate sizing optimization has been emphasized by both industry and academia for a number of years. The 2012/2013 ISPD gate sizing contests [119][120] have given practical impetus to academic research, using industry-standard benchmark data formats and constraint types. The ISPD contest enablement spans consideration of interconnect

parasitics, maximum transition time (*MaxTran*) and capacitance (*MaxCap*) constraints, and use of a commercial signoff timer for timing analysis. Several academic sizers achieve good performance on the contest benchmarks. However, due to highly artificial gate delay/power modeling, as well as the lack of real-world timing constraints, winning codes are unlikely to be able to handle core challenges of gate sizing optimization in modern industrial designs. Indeed, it is our observation that the simplified constraints in contest benchmarks can potentially drive academic sizers in wrong directions. This section describes our experience in identifying and overcoming this mismatch, as we evolved a successful academic sizing approach to perform well in an industry design context.

Limitations of Academic Sizers

We observe that the academic sizing context has several potential inadequacies or gaps with respect to commercial use cases, including the following. While an academic contest will never match the "real world", we wish to highlight gaps that potentially *mislead* academic research efforts' identification of promising directions.

(1) Important constraint scenarios are overlooked in the academic context. Notably, constraints given by academic contests do not include multi-corner multi-mode (MCMM) timing analysis. MCMM timing signoff is essential for modern product designs, where chip designs typically operate (i) under various operating conditions with different temperatures and voltages, (ii) in multiple functional scenarios such as sleep mode and active mode, and (iii) in a regime of manufacturing process variations [172][198]. In digital timing signoff, a *timing corner* represents a particular combination of process, voltage and temperature (PVT) status, and corresponds to a set of timing libraries (Liberty) characterized for that dedicated PVT corner. A *mode* represents a functional scenario, and is characterized by timing constraints (SDC). A *timing view* is defined by a pair of a timing corner and a mode [172][198]. In the MCMM context, multiple timing views are considered, and must be simultaneously comprehended by gate sizing optimization.

In the MCMM context, the electrical properties of each gate, and of its driven net, vary over different corners. We discuss below how achieving and maintaining timing signoff across multiple corners and modes (e.g., nominal, turbo and test modes) seems to require very different optimization strategies

from optimizations that are successful for a single corner/mode. Further, a gate sizing optimization that does not comprehend multiple corners/modes cannot – to our knowledge – achieve a timing-legal solution for all corners and modes. Handling of other constraint types, such as multiple clock/power domains, or timing exceptions, seems to be further removed from the core heuristic design of a gate sizing tool.

(2) Academic sizers are not "robust" across different designs and technologies. Due to the nature of contests, academic sizers can be "over-trained" with the specific set of testcases and objectives that are given by a contest. [87] evaluates a commercial sizer and a contest-winning academic sizer with both contest benchmarks and real designs synthesized with foundry technologies. The academic sizer is observed to perform better on contest benchmarks while showing worse results on real designs, as compared to the commercial sizer. The authors of [87] point out that the change in tools' relative superiority across technologies raises the possibility that the academic sizer might be specialized to the contest benchmark designs.

In our experience, an academic sizer that applies strategies particularly adapted to contest benchmarks may not be *capable* of producing good solutions for real-world designs. The reasons are (i) real designs can differ from contest designs with respect to timing slack distribution, netlist structure, instance count, etc.; and (ii) academic sizers trained for a particular contest technology may not perform well with different technologies since the electrical characteristics (i.e., leakage/dynamic power-delay tradeoff curves of standard cells) can differ meaningfully across process technologies. We have also found that (iii) the simplified and artificial delay/power Liberty model of academic contests cannot capture meaningful electrical attributes of production cell libraries. More specifically, nonlinear and state-dependent power and delay values, nonlinear input capacitance values, and multiple power/ground pins, which are generally seen in foundry Liberty models, are absent from the ISPD contest Liberty models. Due to unrealistic Liberty models, contest benchmarks might not expose core challenges of sizing optimization in product designs. For instance, fixing MaxTran violations in an MCMM context was not comprehended. This creates a risk of misdirecting considerable academic research effort.

(3) Sizing contests typically do not require support for standard formats of Liberty and design files. Examples of standard formats include Verilog netlist (.v), extracted interconnect parasitics (SPEF), and timing constraints (SDC). Typically, contest organizers provide simplified versions of such

files with a parser that only comprehends the simplified input formats. (Yet, open-source and arguably more robust parsers for the full standard formats are available, e.g., [205] [186] [193].) In our view, the lack of (insistence on) support for standard formats unnecessarily hampers transfer of academic sizers to real-world design applications. This necessitates workarounds such as those developed in [87]. Importantly, efforts that chain together the executables of entries from multiple academic contests are blocked from assessment in real-world contexts.

Our Work

In this work, we present key learnings from a multi-year "journey" to make an academic sizing tool applicable to, and yield benefits for, a real industrial IC. We describe aspects of modern industrial designs that are generally not comprehended by academic sizers, but that strongly affect choice of optimization and metaheuristic techniques. We also observe how various optimization techniques used in academic sizers might not be appropriate for product designs due to practical issues such as runtime. We have addressed such gaps between contest-driven research and the real-world application by developing *Sizer*, a new, robust gate sizing optimization tool that incorporates a near-complete change of techniques as compared to our starting point of Trident [84]. The transition from an academic contest setting to real-world designs shows that techniques beneficial in the contest setting may not have benefit for real designs – forcing the development and tuning of a number of new techniques. Ultimately, Sizer achieves 7% leakage power reduction over the commercial tool's high-effort solution on a production design, with signoff at over two dozen mode-corner combinations.

Our contributions are summarized as follows.

- We highlight gaps between academic sizers and commercial sizers due to missing real-world constraints and the characteristics of industrial designs. We describe challenges of transfer/productization that include MCMM timing signoff, MaxTran constraints, hold time constraints, and complex timing structure of a product design.

- We suggest that the "competitive landscape" of the gate sizing optimization – including aspects that are particularly challenging to academic approaches – should be better conveyed to the research

community. Our experiences with Sizer highlight how obliviousness to aspects of real-world application such as practical runtime/memory limits, input-dependent performance models, etc. can easily prevent assessment of an academic sizer within a commercial design context.

- We develop Sizer, a new academic gate sizing tool that is applicable to modern industrial designs. Results reported below show that Sizer achieves solution quality improvement over high-effort commercial tool results, and achieves benefits for a real industrial IC. Sizer embodies a near-total change of techniques as compared to the starting point of Trident [84], which had been successful at the ISPD-2013 gate sizing contest [120].

- We implement dynamic and total power estimations which enable Sizer to go beyond the original contest optimization objective i.e., leakage power-only optimization, and to smoothly control the tradeoff between dynamic and leakage power optimization.

- We compare Sizer with two leading commercial sizers.[1] We achieve 7% leakage power reduction over the high-effort solution of a commercial tool on a design from *NXP Semiconductors* [190]. We also successfully apply Sizer to various testcases synthesized with different foundry technologies.

- We study the impact of various sensitivity functions on the solution quality of Sizer and provide intuition regarding the choice of sensitivity functions.

2.1.1 Related Work

Reflecting the importance of the application, numerous algorithms for both continuous and discrete gate sizing optimizations have been proposed in the literature. Earlier works have focused on continuous gate sizing that optimizes parameters of transistors [43] or of standard cells, such as drive strength, input-pin capacitance, etc. Discrete or library cell-based gate sizing works have applied a wide variety of optimization techniques – Lagrangian relaxation (LR) [26][29][44][69][99][113][121][131][134][136][150][161];

[1]The two commercial sizers are from two commercial entities and are selected from among various commercial sizers (standalone, or integrated in a timing or a place-and-route tool) that are listed under Timing Analysis, ASIC Layout and/or Power Analysis and Optimization by the industry analyst firm *Gary Smith EDA* [177] over the past three years. The universe for this selection includes *Synopsys PrimeTime* [198], *Cadence Encounter Digital System / Innovus* [171][174], *Blaze MO* [200], and *Cadence Tempus* [172]. We are unable to identify the tools more specifically due to license restrictions and sensitivity of EDA vendors.

Table 2.1: Summary of works on gate sizing optimization.

Work	Year	Framework							Objective				Interconnect delay	MaxTran	MaxCap	MCMM	Real designs
		LP	LR	DP	SF	BB	SA	RA	Leakage	TotalPwr	Delay	Area					
[13]	90	✓									✓						
[34]	05	✓								✓			✓				
[77]	09	✓							✓								
[26]	99		✓									✓	✓				
[29]	05		✓						✓								
[69]	11		✓							✓			✓				
[44][99][113]	14, 12, 14		✓						✓				✓	✓	✓		
[121][122]	11, 12		✓									✓	✓	✓	✓		✓
[131][134]	13, 15		✓							✓			✓				✓
[136]	15		✓						✓	✓			✓			✓	
[150]	02		✓								✓		✓				
[161]	15		✓						✓				✓				
[65]	09			✓								✓					
[111]	10			✓								✓	✓				
[130]	11					✓				✓			✓				✓
[133]	13						✓		✓	✓					✓		
[157]	09							✓	✓				✓				
[58]	06				✓				✓				✓				
[66][129]	12, 12				✓				✓				✓	✓	✓		✓
[84]	13				✓				✓			✓	✓	✓	✓		✓
[144][154]	04, 00				✓					✓							
our work					✓				✓	✓			✓	✓	✓	✓	✓

dynamic programming (DP) [65][111]; sensitivity function (SF) [58][66] [84][129][144][154]; branch and bound (BB) [130]; linear programming (LP) [13][34][77]; parallel and randomized algorithm (RA) [157]; and Simulated Annealing (SA) [133]. Table 2.1 taxonomizes previous gate sizing work. Columns 2 – 8 contain the frameworks that are used for each gate sizing approach. Columns 9 – 12 show the objective function of each work. Columns 13 – 16 show the important considerations for real-world gate sizing. The last column shows whether the literature applies its approach to real product designs. An overview of selected recent literature for two popular frameworks, LR and SF, is as follows. Some previous works cannot be categorized into a single particular framework, since more than one frameworks or techniques are used. We attempt to categorize each reference according to the predominant framework or technique used.

LR-based approaches. A number of recent works use LR-based methods for gate sizing optimization. A successful recent application of LR-based gate sizing optimization for industrial designs is described by Ozdal et al. [121][122]. In [121][122], a cost function comprehending the tradeoff between power and timing slack is formulated and then relaxed to a Lagrangian subproblem by Karush-Kuhn-Tucker conditions as in [26]. The subproblem is modeled as a graph problem and solved by using critical tree extraction and DP-based optimization. The work of [121][122] is notable for its thorough treatment of real-world issues such as those we highlight. But, details of implementation are not available to the

research community. Subsequent academic work driven by [119][120] does not capture real-world issues as [121][122] do. Huang et al. [69] suggest a method to obtain Lagrangian multipliers based on the timing history of previous iterations to improve the conventional subgradient-based method. Rahman et al. [131] use an extended logical effort for gate delay modeling to formulate LR problems to which dynamic programming is applied. Flach et al. [44] propose a gate sizing optimization flow that combines LR-based framework and various heuristics. In [44], the LR problem is solved by a sensitivity-based approach; greedy timing and power optimization is subsequently performed. Reimann et al. [134] extend [44] to adapt the LR-based gate sizing to industrial designs. In [134], runtime scalability, preserving timing quality and incremental optimization are considered. Roy et al. [136] solve the gate sizing problem in the context of multiple operating conditions. They extend the LR-based gate sizing approach of [26][121] to support multiple scenarios. More specifically, the authors of [136] use the weighted sum of leakage and dynamic power across different operating conditions as the objective function.

We note that many of LR-based works construct simple *analytic* gate delay models to encompass the discrete gate sizes found in Liberty gate timing libraries. Such analytic gate delay models might not be accurate due to nonlinear characteristics of gate delays. The suboptimality caused by the combination of inaccurate delay models and intrinsic discreteness of the gate sizing problem is one of the major limitations of LR-based approaches. Furthermore, for industrial designs, model-based mathematical methods may be inefficient due to complex constraints such as MaxTran and MaxCap.

SF-based approaches. Wei et al. [154] use the sensitivity function (SF) approach (i.e., iteratively making discrete sizing moves to follow the gradient of a given SF) for simultaneous sizing and dual-V_t assignment. Srivastava et al. [144] propose a dual-Vdd and dual-V_t assignment method based on sensitivity calculations. Gupta et al. [58] use the sensitivities of leakage and delay to gate-length biasing for leakage power optimization. Rahman et al. [129] apply a cost function that considers total slack and leakage changes. Kahng et al. [66] propose sensitivity-guided metaheuristics based on sequential importance sampling and a multistart technique with various sensitivity functions to optimize gate sizing and V_t flavor for minimized leakage. In follow-on work, the optimizer is improved with an efficient and accurate internal timer based on various delay models [84]. To ensure the accuracy of the internal timer, timing information is correlated to the signoff timer's analysis *during* the gate sizing optimization, using techniques proposed

in [117][85].

Our work is distinguished from previous literature in that – to the best of our knowledge – it is the first academic work, available to public [203], that simultaneously addresses all the essential constraints for industrial designs. Further, we provide evaluation using a product design in an industry context.

2.1.2 Constraints for Modern Industrial Designs

We now review examples of practical constraints that are not emphasized in previous published works and academic contests, but are critical for modern industrial designs.

Ozdal et al. [121][122] give a comprehensive summary of the main optimization challenges for gate sizing in modern industrial designs. These challenges are formulated in the ISPD-2012 and the ISPD-2013 Gate Sizing Contests [119] [120]; many practical considerations such as use of a golden signoff timer, interconnect parasitics, maximum transition and capacitance constraints are addressed in the provided testcases. However, in addition to unrealistic Liberty, we find that there are several missing constraints that must be considered for a product design in industry.

Multi-Corner Multi-Mode (MCMM). For modern product designs, a number of PVT corners, along with various functional modes, must be considered during timing signoff. Due to the different delay and power characteristics of cells across multiple corners, it is difficult to achieve a converged solution in the MCMM context. In other words, a signed-off netlist at a particular corner and mode could have timing violations at other corners and modes. Furthermore, there exist "ping-pong" situations where an upsizing move for setup timing recovery in a timing view (a pair of PVT corner and mode) causes timing violations in another timing view. Figure 2.1 shows an example of the ping-pong situation. We assume that C_2 and C_4 are on the setup timing-critical path in $view1$, and that C_1, C_3, C_5 and C_6 are on the timing-critical path in $view2$. To reduce the critical path delay in $view1$, C_2 must be upsized. However, the increased input capacitance of C_2 increases the load capacitance of C_1 (a fanin cell of C_2) and thus increases C_1's delay. As a result, a timing violation occurs in $view2$. Similarly, upsizing cells in $view2$ can create timing violations in $view1$. In modern process technologies, the ping-pong situations become significantly worse with the explosion of PVT corners [82]. A gate sizing optimization must comprehend the timing impact of each V_t swapping or width sizing move in all views simultaneously, and be able to avoid the ping-pong

situations efficiently to obtain a converged solution.

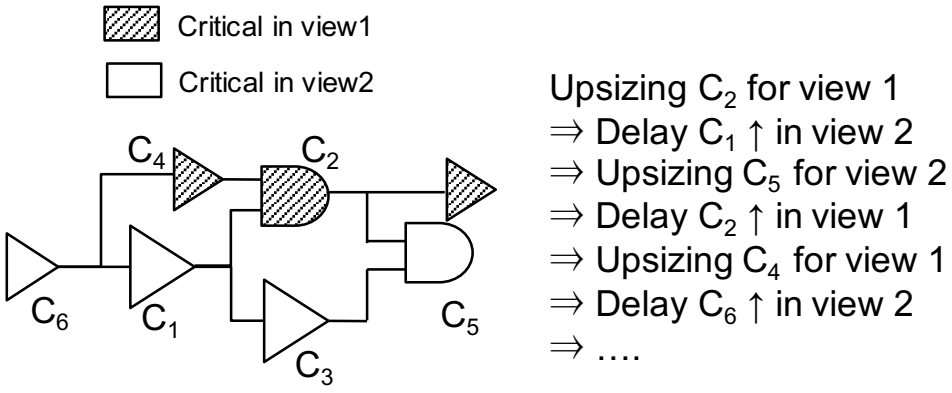


Figure 2.1: Example showing a ping-pong situation during timing recovery for MCMM.

Importance of transition time. The maximum transition (MaxTran) constraint is an upper limit for the transition time at a pin of a gate [119][120][172][198]. The MaxTran constraints are specified in timing libraries or via the set_max_transition command in EDA tools [172][198]. The actual transition time is checked against the MaxTran constraint for every pin in the netlist. For example, if the input pin of an INVX1 instance has a MaxTran constraint of 10, and the actual transition time at the input pin is 15, then the pin violates the MaxTran constraint.

Fixing MaxTran violations is important for product designs for (i) an accurate timing analysis, and (ii) total power optimization. More specifically, in NLDM (Non-Linear Delay Model)-library-based timing analysis, which is still widely used in mature process technologies, violating the MaxTran constraint can induce an inaccurate cell delay calculation due to extrapolation of the cell delay table. A large transition time also increases the internal power of gate instances and harms netlist quality in terms of total power.

The ISPD contest benchmarks [119] [120] include MaxTran constraints as one of the criteria for legal solutions. However, due to the unrealistic timing library, MaxTran violations are relatively easy to fix in the contest technology. For example, the sensitivity of the output transition time of a cell to its load capacitance is relatively small compared to the sensitivity to its drive strength. Thus, in the contest technology, upsizing a cell will always trivially cure a MaxTran violation without causing any new violation on its upstream driving cell. However, in reality, depending on netlist topology (e.g., having many high-fanout nets) and technology, MaxTran constraints can be easily violated and the violations

are difficult to fix. For example, upsizing cells causes transition time violations at the output pins of their fanin cells due to load capacitance increase. Thus, a viable strategy for fixing MaxTran violations must comprehend both the topology of the input netlist and the ripple effect of upsizing cells. We observe that in practice, the MaxTran constraint becomes a "first-class" concern at $65nm$ and below enablements; a gate sizing optimization that is oblivious to such constraints can result in numerous violations, even when other timing constraints such as setup and hold are met.

Hold time consideration. Hold time violations are more critical than setup time violations in product designs since the functionality of a design fails if any hold time violation exists. Despite its importance, the hold time constraint has not been emphasized in academic benchmarks due to the nature of gate sizing for power optimization; most cells are downsized and replaced with higher-V_t cells, which are less likely to incur many hold time violations. However, hold time violations become more critical in the MCMM context, as well as during timing recovery and MaxTran fix optimizations. In particular, upsizing width or decreasing V_t during the timing recovery can lead to hold time violations in a hold-critical timing view, depending on the structure of timing paths in the input design.

Complex structure of a product design. A product design is more complicated than academic benchmarks in many aspects, as it may involve multiple power domains, multiple clocks, and the existence of memories and macros. To support multiple power domains, instances in different power domains must be analyzed with multiple timing/power Liberty tables. Different clock periods due to multiple clocks per timing endpoint such as flip-flops and primary outputs must be handled for an accurate timing analysis. Additional handling in timing analysis is required for memories and macro blocks. That is, the timing graph of a design must properly capture SDC constraints pertaining to timing paths involved with memories or macros. Lack of understanding of memories and macro blocks in an academic sizer results in significantly degraded solution quality with inaccurate timing slack values.

2.1.3 Methodology

We now describe key techniques incorporated in Sizer. Driven by practical constraints seen in product designs, we introduce new features and techniques including (i) MCMM-aware SF; (ii) total power estimation; and (iii) MaxTran violation fixing.

Comparisons to Trident [84]

Trident is an academic sizer originally developed for the ISPD-2013 Gate Sizing Contest [120]. The core engine is based on sensitivity-guided metaheuristics. A multistart technique (*go-with-the-winners*, or GWTW [5][50]), kick moves (following the *large-step Markov chain* optimization technique [116]) and *peephole optimization* are used in Trident. The GWTW metaheuristic performs repeated optimizations within randomized multistarts to effectively explore a large search space. More specifically, Trident runs multiple optimizations concurrently, guided by different sensitivity functions, and records the best solution at prescribed intervals. The kick move technique, which originated in large-step Markov chain optimization, attempts to make a large change, i.e., by upsizing many cells at once, to escape local optima and reach a solution that is closer to a global optimum. The peephole optimization entails exhaustive search for the best size combination for a small path-connected series of cells.

Table 2.2 summarizes the optimization techniques used in Trident ([84]) and Sizer (our work). We do not include the sensitivity function (SF) framework itself in the table since SF-based gradient-following (greedy, steepest descent) algorithm is not new to [84]; it was in the SensOpt package [202] from which [84] was derived. The Blaze MO tool [200] is another example of sensitivity-based sizer. MCMM-awareness is our key improvement to support product designs. Data structures in Sizer store timing information for each timing view. New sensitivity functions for MCMM are developed to enable effective MCMM-aware optimization for both timing recovery and power reduction. Total power estimation is added as well, to guide the total power optimization. Additionally, with the increased importance of transition time in real designs, we notice that obtaining accurate transition time is essential for the optimization. We thus implement the internal timer of Sizer to improve the accuracy of transition time. We adopt the transition time correlation method of [85], whereas Trident [84] performs only slack correlation.

We note that a number of distinguishing innovations in [84] such as the use of GWTW and peephole optimization were eventually discarded in Sizer because no benefit could be seen in the production context. For realistic designs with many (i.e., $>$30) timing views and multiple clocks, the optimization complexity increases dramatically. For example, the GWTW paradigm requires additional cores according to the number of multistarts, and each individual optimization requires multiple timers to support MCMM. Thus,

GWTW demands huge computing resources in the production context, due to its greater complexity of timing analysis. Peephole optimization is also computationally demanding, since it attempts to evaluate multiple combinations of solutions for several cells at a time. Such approach is infeasible if computing resources and runtime are limited. Crucially, our background experiments showed that even with huge computing resource consumption, the two techniques have little or no benefit in terms of solution quality. For example, the additional leakage reduction obtained from GWTW is 1% at the cost of 2.5× runtime, for the AES design. For the M0 design, 3% more leakage reduction can be achieved, but at the cost of 4.3× runtime. With use of peephole optimization, our background studies found that the final leakage power worsens for both the *AES* and *M0* designs, by 2.6% and 1.0%, respectively.

Table 2.2: Comparisons of Trident [84] (Tri) and this work (Szr).

Techniques	**Tri [84]**	**Szr**
MCMM-aware static timing analysis	-	✓
MCMM-aware timing recovery	-	✓
MCMM-aware sensitivity function	-	✓
Transition time correlation	-	✓
Total power estimation	-	✓
Dynamic change of sensitivity function	-	✓
Prioritization of moves	-	✓
Kick move	✓	✓
Go-with-the-winner (GWTW)	✓	-
Peephole optimization	✓	-

New Sensitivity Functions in the MCMM Context

Sensitivity functions (SFs) are used as guidance to select sizing *moves* (V_t swapping or sizing) that give the maximum power reduction benefit with the minimum timing impact.[2] For this purpose, an SF should calibrate the power benefit at the expense of timing slack degradation, or vice versa. However, it is not straightforward to estimate timing or power impact of a sizing move on the design since (i) the entire timing graph, which is very complex in modern product designs must be comprehended due to the nature of timing calculation propagation toward downstream cells, and (ii) in the MCMM context, variations across multiple corners must be considered as discussed in Section 2.1.2 above.

[2]Throughout this work, a "move" indicates a V_t-swapping or sizing move of a cell, not a physical movement in placement.

To address these challenges, we introduce new estimations of timing and power impact, considering MCMM, for our SFs as follows.

Table 2.3: Notations.

Notation	Meaning
P	weighted sum of leakage and dynamic power
D	clock period
TNS	total negative setup time slack
WNS	worst negative setup time slack
s_i	setup time slack of cell i
d_i	delay of cell i
c_i	load of cell i
$tran_j$	transition time at pin j
$tran_{max}$	maximum transition time
s_i^{tran}	maximum transition time slack of cell i
$\#paths_i$	number of register-to-register paths going through cell i
β_v	weighting factor for view v for MSF3
γ	normalizing factor for transition time slack

Table 2.3 summarizes the notations that we use in our SFs. Each notation can be extended by adding subscript v to represent the notation associated with timing view v. I.e., $s_{i,v}$ is slack of cell i in timing view v.

- P denotes a weighted sum of leakage and dynamic power. We provide a detailed discussion of power calculation below.

- s_i denotes the timing slack of cell i. We consider only setup time slack for our SFs. We consider hold time differently in our optimizer. We revert any V_t-swapping/sizing move that results in any hold time violation. This is for two main reasons. First, during the power reduction stage, due to the nature of power optimization where cells are downsized and/or swapped to higher V_t cells, delays of the optimized cells typically increase, which is actually better for timing paths with respect to hold time. Second, during timing recovery, as we seek to cure setup-violating paths, the target cells for upsizing are mostly setup-critical.[3] Also, inclusion of hold time increases computation complexity.

[3]Even in industrial timing ECO flows, the setup-hold critical paths are typically fixed by engineers manually instead of commercial sizers [128].

- d_i denotes the delay of cell i. For Δd_i in SFs, we calculate the delay of each input to output timing arc of cell i before and after a given move, and return the largest arc delay change.
- c_i is the output load of cell i; this output load includes wire capacitance and the sum of input pin capacitances of cell i's fanout cells.
- s_i^{tran} is the minimum value of $(tran_{max} - tran_j)$, over all pins j of cell i.
- $\#paths_i$ is the number of register-to-register paths that pass through cell i. $\#paths_i$ is calculated by multiplying the number of downstream registers and the number of upstream registers of cell i.

Table 2.4: Sensitivity functions for a single view and MCMM.

Type	Index	Equation
Single	SF1	ΔP
	SF2	$\Delta P \cdot s_i$
	SF3	$\Delta P / \Delta d_i$
	SF4	$(\Delta P \cdot s_i)/\#paths_i$
	SF5	$(\Delta P \cdot s_i)/(\Delta d_i \cdot \#paths_i)$
	SF6	$-\Delta P/(\Delta s_i \cdot \#paths_i)$
MCMM	MSF1	$\Delta P \cdot \min(\min_{v \in views}(s_{i,v} - \Delta d_{i,v}), \gamma \cdot \min_{v \in views} s_{i,v}^{tran})$
	MSF2	$(\Delta P \cdot \min(\min_{v \in views}(s_{i,v} - \Delta d_{i,v})/c_{i,v}, \gamma \cdot \min_{v \in views} s_{i,v}^{tran}/c_{i,v}))$
	MSF3	$\Delta P \cdot \#paths_i \cdot sum_{v \in views} \beta_v / \Delta d_{i,v}$

Table 2.5: Parameter conditions and their implications.

Condition	Parameter Condition				Implication
	ΔP	s_i	Δd_i	Δs_i	
1	+/-	+/-	+	+	N/A
2	+/-	+/-	-	-	
3	-	+/-	-	+	Always pick
4	+	+/-	+	-	Always avoid
5	+	-	-	+	Timing recovery (TR)
6	+	+	-	+	Avoid (Aggressive TR)
7	-	+	+	-	Power reduction (PR)
8	-	-	+	-	Avoid (Aggressive PR)

Table 2.4 shows SFs for single-view and MCMM optimizations. ΔP, Δs_i and Δd_i denote the differences in P, s_i and d_i values, respectively, before and after a move of cell i. For example, ΔP is the power (total or leakage power) of the design after a move subtracted by the original power value.

SF for a single view. In Table 2.4, SF1 simply prioritizes cells that offer large power reduction. SF2 prioritizes cells with large timing slacks that also offer large power reduction. SF3 selects cells that have a smaller delay penalty but a larger power reduction. SF4 is a variation of SF2, but we add $\#paths$ so that we do not pick cells that affect slacks of many timing paths. SF5 is a combination of SF3 and SF4. SF6 picks cells that have less impact on total negative slack (TNS) but have more power reduction. To reduce runtime, we estimate ΔTNS as $(-\Delta s_i \cdot \#paths)$.

SF for MCMM. Simultaneous consideration of multiple timing views is important since (i) power-critical views are different from timing-critical views, and (ii) timing must be signed off in all the given timing views. Thus, SFs always need to consider the "worst" timing/power impact considering each of the given timing views. We propose new SFs to guide the optimizations in the MCMM context, i.e., MSF1, MSF2 and MSF3, as shown in Table 2.4. Here, ΔP in all MSFs is calculated in the most power-critical view. MSF1 prioritizes cell moves that lead to large setup time slack ($s_i - \Delta d_i$) and MaxTran slack, in conjunction with large power reduction. For the setup time slack and MaxTran slack, the minimum values across all timing views are considered. For γ, normalizing factor for MaxTran slack, we empirically use 0.5 in our reported studies based on results of background experiments. MSF2 is a variation of MSF1. In MSF2, c_i is used as a denominator to avoid downsizing cells with larger load capacitance (or many fanout cells). This helps to avoid MaxTran violations.

MSF3 uses weighting factor β_v to prioritize timing-critical views and thus address different timing constraints and slacks for each timing view. We study three weighting factors: (i) $\frac{1}{D_v}$; (ii) $\frac{TNS_v}{D_v}$; and (iii) $\frac{WNS_v}{D_v}$. Based on our experimental results on design M0 in $28nm$ LP technology, using weighting factors (i), (ii) and (iii) respectively leads to 26%, 13% and 4% less runtime needed to achieve a timing-feasible solution, compared to timing recovery without weighting factors.

Example. The following example illustrates how SF values for a single timing view and MCMM are calculated. Let us assume three timing views, i.e., v_1, v_2, v_3, and v_3 is the power-critical view. The corresponding power, setup time slack, delay, transition time slack, $\#paths$ and output load for cell i are summarized as follows.

- Setup time slack : $s_{i,1} = 10$, $s_{i,2} = 5$, $s_{i,3} = 20$

- Transition time slack : $s_{i,1}^{tran} = 20$, $s_{i,2}^{tran} = 16$, $s_{i,3}^{tran} = 40$
- ΔPower : $\Delta P_1 = 3$, $\Delta P_2 = 4$, $\Delta P_3 = 6$
- ΔDelay : $\Delta d_1 = 6$, $\Delta d_2 = 3$, $\Delta d_3 = 1$
- The number of paths : $\#paths = 5$
- Output load : $c_{i,1} = 2$, $c_{i,2} = 2$, $c_{i,3} = 2$
- β : $\beta_1 = 0.3$, $\beta_2 = 0.2$, $\beta_3 = 0.1$

Each SF considering view v_1 only is calculated as follows.

- SF1: $\Delta P_1 = \Delta P = 3$
- SF2: $3 \cdot 20 = 60$
- SF3: $3/6 = 0.5$
- SF4: $3 \cdot 20/5 = 12$
- SF5: $3 \cdot 20/(6 \cdot 5) = 2$
- SF6: $-3/(20 \cdot 5) = -0.03$

And, each MSF considering all timing views, i.e., v_1, v_2 and v_3, is calculated as follows.

- MSF1: $\Delta P = \Delta P_3 = 6; \min_v s_{i,v} - \Delta d_{i,v} = 2; \min_v s_{i,v}^{tran} = 16; MSF1 = 6 \cdot \min 2, 8 = 12$
- MSF2: $\Delta P = \max_v \Delta P_v = 6; \min_v (s_{i,v} - \Delta d_{i,v})/c_{i,v} = 1; \min_v s_{i,v}^{tran}/c_{i,v} = 8; MSF2 = 6 \cdot \min 1, 4 = 6$
- MSF3: $6/5 \cdot (0.3/6 + 0.2/3 + 0.1/1) = 0.26$

The complexity of SF and MSF computation for a single cell are O(1) and O(N), respectively, where N is the number of timing views.

Parameter conditions for SF usage. Table 2.5 summarizes parameter conditions and their implications. "+" means a positive value, "-" means a negative value. Δd_i and Δs_i cannot be both

positive or negative, so we do not consider such cases (Conditions 1 and 2). If fanin and/or fanout cells of target cell i are affected, Δd_i and Δs_i can be both positive or both negative. However, such cases are extremely rare in our experiments. We always pick moves that provide both power and timing benefits (Condition 3). Similarly, we avoid moves leading to both power and delay penalties (Condition 4). For timing recovery, we use SF6 and MSF3 if Condition 5 is met. We do not consider cells in Condition 6 for upsizing or decreasing V_t. During power reduction, we calculate SFs for a move such that Condition 7 is met. SF1-6, MSF1-2 are used for such cases in our experiments. For Condition 8, we do not allow downsizing/increasing V_t since aggressive power reduction leads to excessive timing violations.

Total Power Estimation

In this section, we describe Sizer's capability to perform total (weighted) power reduction. To comprehend the total power optimization, we consider both leakage and dynamic power. Dynamic power consists of net switching power and cell internal power.

For leakage power calculation, we directly use the leakage value of each cell from Liberty files. Since leakage is state-dependent in the Liberty files, we use the average leakage value across different states as an approximation. We empirically observe that such approximation overall improves optimization runtime while offering relatively accurate leakage estimation. If state probabilities are available from dynamic simulation and power analysis, then replacing the arithmetic mean across states with a weighted average would be straightforward.

We estimate the net switching power of a net as $\frac{CV^2Tr}{2}$, where C is the total net capacitance, i.e., the sum of wire capacitance and the capacitance of cell input pins connected to the output of the wire, V is the supply voltage, and Tr is the toggle rate of the net. We extract the toggle rate and wire capacitance of each net from a golden signoff timer; input pin capacitances of cells are obtained from Liberty files.

For cell internal power, we obtain the values from lookup tables (LUTs) in Liberty files based on input slew and output load of the cell. We note that there exist multiple internal power LUTs for a given cell, where each table corresponds to a particular transition arc, that is, a pair of an input pin and an output pin. As our power analysis is vectorless, we normalize the internal power values related to different input pins based on the toggle rates at pins of the cell. An example of a 2-input cell is given in Figure 2.2, where

cell G has input pins A and B. We estimate the internal power of cell G (P_G) as follows.

$$P_G = \frac{(P_{int,A} \cdot Tr_A + P_{int,B} \cdot Tr_B) \cdot Tr_Z}{Tr_A + Tr_B} \tag{2.1}$$

Here, (i) $P_{int,A}$ and $P_{int,B}$ are respectively the internal power values related to input pins A and B based on LUTs, and (ii) Tr_A, Tr_B and Tr_Z are respectively the toggle rates at input pins A, B and output pin Z. In the example, P_G is calculated as 0.67 according to Equation (2.1).

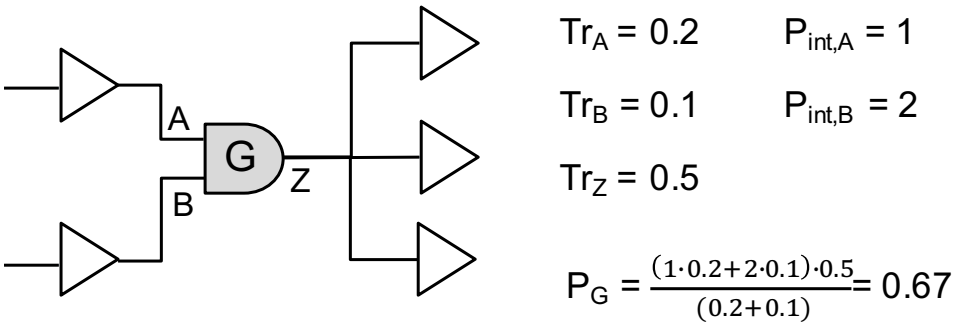


Figure 2.2: Example internal power estimation of a 2-input cell.

Note that for every sizing or V_t swapping, pin capacitance and slew values change and thus the net switching power and internal power change. To adapt our sensitivity functions for total power optimization, we replace the power terms P in SFs above with $\alpha \cdot P_{leak} + (1 - \alpha) \cdot (P_{int} + P_{sw})$, where P_{leak}, P_{int} and P_{sw} indicate leakage power, internal power and net switching power respectively. α is a weighting factor to trade off between leakage power and dynamic power optimization. A larger (resp. smaller) α value leads to a greater reduction in leakage power (resp. dynamic power). To our knowledge, previous literature on gate sizing optimization for total power optimization typically uses gate capacitance or transistor width for dynamic power estimation. By contrast, to achieve a more accurate total power estimation, we follow industrial standard practice and estimate dynamic power based on both net switching power and cell internal power.

Handling Maximum Transition Constraints

Our Sizer has two specific features to handle MaxTran violations: (i) transition time correlation, and (ii) a dedicated procedure for fixing MaxTran violations.

As noted above, Trident [84] performs slack correlation between its internal timer and the signoff timer for a more accurate slack estimation. However, lack of transition time correlation in its optimization can result in additional MaxTran violations due to transition time mismatches. We thus perform transition time correlation in Sizer using the method of [85], in addition to the slack correlation.

Algorithm 1 Procedures for fixing MaxTran violations.

Procedure: $FixMaxTran(netlist, max\_tran)$
Input: netlist $netlist$, MaxTran constraint max_tran
Output: optimized netlist with reduced transition time violations

for all cell $c \in netlist$, in topological order **do**
  for all violating pin $vp \in c$ **do**
    if vp is an output pin of c **then**
      for all $focell \in$ fanout cells of c **do**
        downsize $focell$
        if $s_c < 0$ **then** revert the move
        if $tran_{vp} \leq max\_tran$ **then break**
      end for
      while $tran_{vp} > max\_tran$ **do**
        UpsizeCellforTran(vp, c)
      end while
    else
      $ficell \leftarrow$ the fanin cell of the input net of vp
      while $tran_{vp} > max\_tran$ **do**
        UpsizeCellforTran(vp, $ficell$)
      end while
    end if
  end for
end for

Procedure: *UpsizeCellforTran*(vp, $cell$)
Input: violating pin vp, cell $cell$
Output: updated netlist

$SF\_size \leftarrow \Delta tran_{vp}/\Delta P$ (when upsized)
$SF\_vt \leftarrow \Delta tran_{vp}/\Delta P$ (when V_t swapped)
if $SF\_size == 0$ and $SF\_vt == 0$ **then return**
if $SF\_size < SF\_vt$ **then**
  upsize $cell$
else
  decrease V_t of $cell$
end if
if $s_{cell} < 0$ **then** revert the move

Algorithm 1 shows procedures to fix MaxTran violations. In the $FixMaxTran$ procedure, if the violation occurs at an output pin, we visit all fanout cells of the violating pin ($focell$), and perform downsizing on each of the fanout cells until the maximum transition violation is fixed, unless the downsizing move leads to a timing violation (Lines 4–8). If downsizing moves of the fanout cells do not fix the maximum transition violation, we perform upsizing or decreasing V_t of the violating cell by invoking the *UpsizeCellforTran* procedure (Lines 9 – 11). If the violation occurs at an input pin, we perform upsizing width or decreasing V_t of the fanin cell by invoking the *UpsizeCellforTran* procedure (Lines 14 – 16). In *UpsizeCellforTran*, we first evaluate the resultant transition time reductions from upsizing and decreasing V_t as well as the corresponding power penalties (Lines 20–21). If the transition time does not improve with both moves, we do not perform upsizing (Line 22). Between upsizing and decreasing V_t, we pick the move that has a larger ratio of transition time reduction to power penalty (Lines 23–25). We revert the move if it leads to a timing violation (Line 28).

Other Techniques for Better Optimization

In this section, we describe several other techniques used to achieve improved optimization.

Prioritization of move type. We introduce prioritization of a certain move type on top of the SF-based prioritization for the following two reasons. First, the types of SFs we use cannot differentiate between (i) a move that leads to large power reduction as well as large delay increase, versus (ii) a move that has small impact on both power and delay. Second, at early optimization stages when there is sufficient timing slack for power optimization, (i) is preferred to (ii), in order to achieve faster convergence to the local minima.

Figure 2.3 shows the leakage-delay tradeoff of buffers (with two V_t flavors and various gate sizes) in (a) $140nm$ and (b) $28nm$ LP technologies. The leakage and delay values of higher (resp. lower) V_t cells are shown in red (resp. blue) dots. The delay values of each gate are calculated based on NLDM libraries with a fixed input transition $50ps$ and a load equal to four times the input pin capacitance of the driving gate. Figure 2.3 shows that the leakage values of higher V_t cells (i.e., UVT and RVT) are lower than those of lower V_t cells (i.e., SVT and LVT). The naming conventions for high and low V_ts vary according to particular foundries and technology nodes. In the $140nm$ dual-V_t library, UVT (Ultra-high threshold

voltage) and SVT (Standard threshold voltage) respectively correspond to high and low V_ts. In the $28nm$ dual-V_t library, RVT (Regular threshold voltage) and LVT (Low threshold voltage) respectively correspond to high and low V_ts.

V_t flavor has much larger impact on leakage power than does gate size. We therefore expect that V_t swapping is a stronger lever than sizing for leakage optimization. However, if we use SF3, move2 in Figure 2.3 will be selected over move1, even though move2 is a stronger solution for leakage optimization. For dynamic power optimization, due to its direct impact on total design capacitance, sizing is preferred over V_t swapping. Thus, in the contest of different optimization objectives, we prioritize either V_t swapping or sizing during our optimization. More specifically, we pick the moves with preferred move type (i.e., V_t swapping or sizing) among the top-k candidate moves (sorted by the sensitivity scores[4]) to execute. Put another way: we ignore the non-preferred move type during our optimization unless there is no move of the preferred move type among the top-k candidate moves. Note that the value of k trades off the effect of sensitivity function versus that of the preferred move type for selection of optimization moves (e.g., a small k indicates that the optimization honors sensitivity scores more). Based on our separate study, we empirically use $k = 40$ (PR_{max} in Table 2.6) in our experiments.

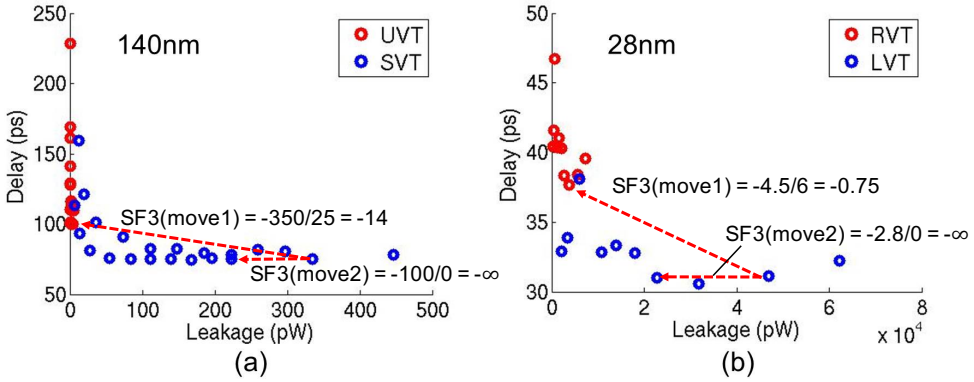


Figure 2.3: Leakage versus delay of V_t swapping and sizing in (a) $140nm$ (TT, $1.5V$, $25°C$) and (b) $28nm$ LP (TT, $1.0V$, $25°C$).

Dynamic change of SF. In a SF-based optimization, the definition of SF can significantly affect the optimization solution quality. An optimal SF choice is dependent on the status of netlists as well

[4]We use "sensitivity score" to refer to the values calculated by SF.

as input design information that is initially given (e.g., foundry technologies or libraries, initial netlists, timing constraints, and optimization objectives). Thus, using a single SF throughout the entire optimization procedure increases the likelihood of getting stuck at local minima. To avoid such a situation, we dynamically change the SF during our optimization to adapt to the status of the input netlist. More specifically, Sizer changes SF whenever the optimization with the current SF cannot achieve further power reduction. Based on our experimental results in foundry $140nm$ technology, such a dynamic change of SF achieves $\sim 2\%$ further leakage power reduction.

Unsuccessful Techniques

In this section, we describe several techniques that have not shown noticeable benefits in our experiments and were eventually dropped.

Tabu search. We have attempted a form of Tabu search [49] during our gate sizing optimization to avoid being stuck at a local minimum. More specifically, we record three to five most recent sizing/V_t-swapping moves performed in the power reduction (resp. timing recovery) stage and forbid these moves for the following timing recovery (resp. power reduction) stage. We initially thought that this would reduce the likelihood of being stuck at a local minimum. However, the Tabu-search-inspired strategy has not helped to improve the solutions for our testcases since there were not many cells that went back to their previous status. We believe that the use of different SFs in the power reduction and timing recovery stages might help avoid previously-visited solutions for each cell.

Smart guardband. We have attempted "smart", or variable guardband-based, optimization. Here, we refer to a timing margin that is added to timing slack as a guardband. Using a positive (resp. negative) guardband means that timing analysis becomes optimistic (resp. pessimistic) by the amount of the guardband. We have tried varying these guardbands depending on the iteration count within the overall optimization. More specifically, we have used a positive guardband at the initial several iterations of the optimization procedure. We then gradually decrease the guardband at the later iterations of the optimization procedure; in the end, zero guardband is applied so that no timing violation is allowed. In our experiments, this approach has not improved the solution but has only rather increased runtime.[5] Due to

[5]This recalls the metaheuristic paradigm of *threshold acceptance*; see, e.g., [64].

the optimism introduced at the first few iterations, Sizer downsizes and/or V_t-swaps gates too aggressively and ends up spending a lot of time on timing recovery.

Multi-step sizing and V_t swapping. We have attempted multi-step sizing and V_t swapping during timing recovery stages. For example, if $1\times$, $2\times$ and $4\times$ sizing options are given for a specific gate instance, we have considered $1\times$ to $4\times$ (two-step) sizing in addition to $1\times$ to $2\times$ (one-step) sizing. Similarly, if LVT, RVT and HVT V_t flavors are given, we have considered LVT to HVT swapping. In our studies, we calculated sensitivity scores for multi-step moves and added them to the candidate list. However, multi-step moves were rarely picked based on the sensitivity scores. Rather, calculating sensitivity scores of these additional moves increases runtime significantly. We conclude that one-step moves are sufficient to consider for timing recovery in our experience.

Overall Optimization Flow

In this section, we describe the overall optimization flow of Sizer and the details of key procedures. Figure 2.4 shows the overall optimization flow of Sizer. The optimization flow is an adapted version of [84] with our new techniques that are essential for commercial product designs. The optimization consists of three stages: (i) the sensitivity-based power reduction stage, (ii) the timing recovery stage, and (iii) the kick move stage. In the power reduction stage, Sizer attempts to downsize cells or swap cells to high-V_t cells based on sensitivity scores until timing becomes infeasible in any timing view. In this stage, the dynamic change of SF, the MCMM-aware sensitivity function and the prioritization of moves are applied as discussed above. In the timing recovery stage, upsizing and/or V_t swapping based on SF is performed to fix any timing violation occurred during the power optimization stage. In the kick move stage, Sizer attempts to upsize/swap cells in order to escape local optima and increase timing slack so that more cells can be downsized. We iterate the optimization loop multiple times and store the best solution (i.e., minimum power solution with legal timing) obtained during the optimizations.

Overall optimization. We have several user-defined input parameters that can tune the optimization flow. Table 2.6 shows the user-defined input parameters and default values. The details of the overall optimization flow are shown in Algorithm 2. We first store the initial solution to *best_sol* to set it as our starting point (Line 1). We set SF to the input SF_p for power recovery procedure (Line 2). We then

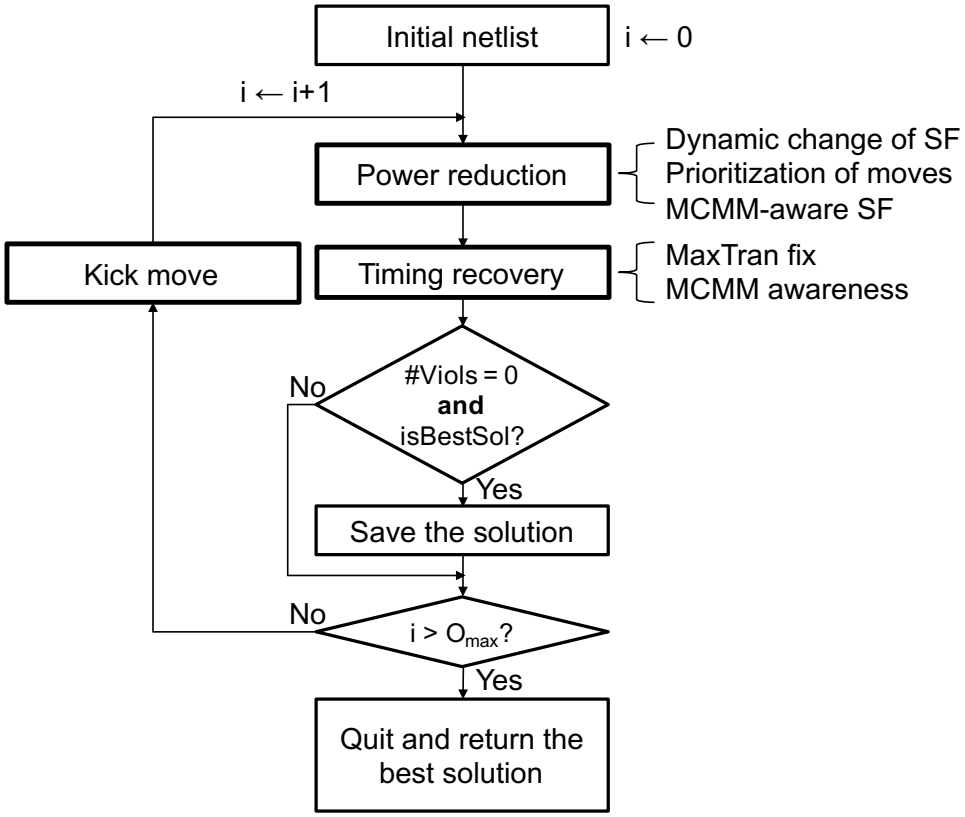


Figure 2.4: Overview of the overall optimization flow.

Table 2.6: User-defined input parameters.

Notation	**Meaning (default)**
SF_p, SF'_p	sensitivity functions for power reduction (MSF1, MSF2)
SF_t	sensitivity function for timing recovery (MSF3)
X	#trials for cell V_t swapping or sizing (20% of total #cells)
α	controls leakage and dynamic power optimizations (0.0)
O_{max}	number of optimization loops (8)
T_{max}	maximum number of iterations of timing recovery (30)
PR_{max}	maximum number of candidate moves for prioritization (40)
S_{max}	threshold for dynamic change of SF (5)
X_k	maximum #trials allowed for kick move (0.05% of total #cells)
th_p	minimum allowed slack threshold for power reduction (-20ps)
th_k	maximum allowed slack threshold for kick move (20ps)
δ	threshold for correlation (5% of total #cells)

perform power reduction with SF, the trial rate X and th_p (Line 5). We give a small negative slack margin th_p as default to allow aggressive downsizing and V_t swapping for power reduction. We observe from our experiments that optimization with a negative slack margin followed by timing recovery typically achieves

Algorithm 2 The overall optimization flow.

Procedure: $PowerOpt(netlist, ...)$
Input: netlist $netlist$, user-defined inputs (See Table 2.6)
Output: an optimized $netlist$

1: $best\_sol \leftarrow input$
2: $SF \leftarrow SF_p$
3: **for** i = 1 to O_{max} **do**
4: $stuck\_count \leftarrow 0$
5: $PowerReduction(SF, X, th_p)$
6: **for** j = 1 to T_{max} **do**
7: **if** no timing violation exist **then break**
8: *TimingRecovery(SF_t,0,X)*
9: **end for**
10: **if** no timing violation **and** power is improved with $current\_sol$ **then**
11: $best\_sol \leftarrow current\_sol$
12: decrease th_k, X_k
13: **else**
14: $stuck\_count \leftarrow stuck\_count + 1$
15: increase th_k, X_k
16: **end if**
17: $KickMove(th_k, X_k)$
18: **if** $stuck\_count > S_{max}$ **then**
19: $SF \leftarrow SF'_p$
20: **end if**
21: **end for**

better solution quality compared to the conventional zero-margin optimization. After the power reduction step, we iteratively perform the procedures for fixing timing violations until all violations are fixed or $j = T_{max}$. If there is no timing violation with the current solution, and its power is less than the best power value, we store the current solution as the best solution (Lines 10–11). And, th_k and X_k are decreased so that less cells are upsized in the kick move stage (Line 12). If not, we increase the $stuck\_count$ which indicates that the optimization gets stuck (Line 14). And, we increase th_k and X_k (Line 15) so that more cells are upsized in $KickMove$. We then perform $KickMove$ (Line 17). If $stuck\_count$ is larger than S_{max}, SF is changed to SF'_p (Line 19). We iterate the optimization loop by O_{max}. In our current implementation, the cells and moves in the $KickMove$ procedure are determined by a given SF, th_k and X_k (Algorithm 5). The SF is deterministic, while th_k and X_k change throughout the entire optimization process. And, although we change th_k and X_k throughout the optimization as described in Algorithm 2,

the values are not randomly selected. Therefore, our current $KickMove$ implementation is not completely stochastic. However, as the parameters th_k and X_k, and the slacks of cells, change throughout multiple iterations, the target cells for V_t swap or sizing will change. The net result is a kick move whereby the current state of the circuit is "perturbed" in a manner designed to escape the current local minimum – sometimes referred to as a *basin of attraction* – and thus avoid cycling in the optimization.

Power reduction. Algorithm 3 illustrates the sensitivity-based power reduction procedure ($PowerReduction$). The inputs to $PowerReduction$ include the sensitivity function SF_p, trial rate X and the slack threshold th_p. We first build a list of candidate solutions where a solution sol indicates a (target cell $sol.cell$, target move $sol.move$) pair, together with its sensitivity score $sol.score$. We calculate the sensitivity score for each (cell, move) pair (Line 4). If ΔP of the candidate (cell, move) pair is negative (indicating that power decreases), we add the solution to the solution list sol_list (Line 5). As we downsize or increase V_t, the delay of the target cell increases while the power decreases in most cases. Thus, the sensitivity score is typically a negative value, where a smaller value (i.e., a larger absolute value) indicates a better solution. Therefore, the solution list is sorted in increasing order of sensitivity scores (Line 8). The only case where the sensitivity score can be positive is when both delay and power decrease. In this case, we set the sensitivity score to the negative infinite number so that we can always pick such moves first. For details, please see Table 2.5.

We attempt each candidate move in the solution list (Lines 10–25). To prioritize V_t swapping (resp. sizing) for leakage (resp. dynamic) power optimization, we search prioritized moves among the first PR_{max} solutions in the solution list (Lines 11–12). In a leakage-only optimization, if we find a V_t-swapping move, we pick the corresponding solution (Line 12), otherwise we pick the first solution among sol_list (Lines 13–14). We then commit the move (Line 16). The committed move is removed from the solution list (Line 17). We then update design timing and check the updated slack of the optimized cell in all timing views. During the timing update process, we periodically correlate timing with a signoff timer for accuracy (Line 18). We revert the move if there is any setup timing (i.e., with respect to th_p), MaxTran or MaxCap violation (Lines 19–20). Even though we check timing for every move to avoid violations, the mismatch between the internal timer and the signoff timer can result in timing violations during our optimization. In this case, we quit the procedure (Line 24) and call the $TimingRecovery$ procedure. The

trials for sizing or V_t-swapping continue until the number of trials reaches X (Line 10).

Algorithm 3 Procedure of power reduction.

Procedure: $PowerReduction(SF_p, X, th_p)$
Input: sensitivity function SF_p, trial rate X, slack threshold th_p
Output: an updated netlist with less power

1: **for all** cell $c \in netlist$ except for clock **do**
2: **for all** move $m \in$ candidate moves M_c **do**
3: $sol.cell \leftarrow c$; $sol.move \leftarrow m$
4: $sol.score \leftarrow CalculateSensitivity(c, m, SF_p)$
5: **if** $\Delta P < 0$ **then** $sol\_list \leftarrow sol\_list \bigcup \{sol\}$
6: **end for**
7: **end for**
8: sort sol_list in order of **increasing** sensitivity score
9: $cnt \leftarrow 0$
10: **while** $cnt < X$ **do**
11: **if** any prior move exists in top PR_{max} solutions **then**
12: $sol \leftarrow PickFirstPriorMove(sol\_list)$; $PR_{max} \leftarrow PR_{max} - 1$
13: **else**
14: $sol \leftarrow sol\_list.front()$
15: **end if**
16: commit sol
17: $sol\_list \leftarrow sol\_list \setminus \{sol\}$
18: **if** $mod(cnt, \delta) = 0$ **then** $TimingCorr()$
19: **if** $s_{sol.cell} < th_p$ **or** new MaxTran/MaxCap occurs in any timing view **then**
20: revert sol
21: **else**
22: $cnt \leftarrow cnt + 1$
23: **end if**
24: **if** $worst\_slack < th_p$ **then break**
25: **end while**

Timing recovery. Algorithm 4 illustrates the $TimingRecovery$ procedure. In the $TimingRecovery$ procedure, we upsize or swap a cell to a lower-V_t cell to fix timing violations caused during the $PowerReduction$ procedure. Fixing MaxTran and MaxCap violations is performed first (Lines 1–2). We fix MaxTran and MaxCap violations before performing the SF-based upsizing/V_t-swapping, since we observe that fixing maximum transition and capacitance violations also helps to fix setup time violations by improving gate delays in our experiments. This is because large transitions and/or load capacitance tend to increase gate delays. We also note that it is recommended to first fix electrical rule violations (e.g., maximum transition and capacitance violations) in the DAC Knowledge Center article of MacDonald

Algorithm 4 Procedure of timing recovery.

Procedure: $TimingRecovery(SF_t, th, X)$
Input: sensitivity function SF_t, slack threshold th, trial rate X
Output: an updated netlist with an improved worst slack

1: **if** MaxCap violations exist **then** *FixMaxCap()*
2: **if** MaxTran violations exist **then** *FixMaxTran()*
3: **for all** cell $c \in netlist$ except for clock **do**
4: **if** $s_c > th$ **continue** // skip this cell
5: **for all** move $m \in$ candidate moves M_c **do**
6: $sol.cell \leftarrow c$; $sol.move \leftarrow m$
7: $sol.score \leftarrow CalculateSensitivity(c, m, SF_t)$
8: **if** $\Delta d_{sol.cell} < 0$ **then** $sol\_list \leftarrow sol\_list \bigcup \{sol\}$
9: **end for**
10: **end for**
11: sort sol_list in order of **decreasing** sensitivity score
12: $cnt \leftarrow 0$
13: **while** $cnt < X$ **do**
14: $sol \leftarrow sol\_list.first()$
15: $orig\_slack \leftarrow s_{sol.cell}$
16: commit sol
17: $sol\_list \leftarrow sol\_list \setminus \{sol\}$
18: **if** $mod(cnt, \delta) = 0$ **then** $TimingCorr()$
19: **if** $s_{sol.cell} < orig\_slack$ **or** creates hold violations **then**
20: revert sol
21: **else**
22: $cnt \leftarrow cnt + 1$
23: **end if**
24: **end while**

[115]. For each non-clock cells with slacks less than th (Lines 3–4), we calculate sensitivity scores (Lines 6–7) for its candidate moves (i.e., one-step upsizing or one-step decreasing V_t). If the $\Delta d_{sol.cell}$ decreases, we add sol to sol_list. We then sort the solution list in decreasing order of sensitivity scores (Line 11).[6] For each solution in the solution list, we commit the corresponding move and update timing (Line 16). During the procedure, we periodically correlate timing with a signoff timer for accuracy, similarly as in the power reduction procedure (Line 18). We then check whether the committed move degrades slack and creates a hold violation in the given timing view v (Line 19). If so, we revert the move (Line 20).

[6]The delay terms (i.e., Δd_i, $-\Delta s_i$) are negative values, and the ΔP term is a positive value since delay decreases and power increases with upsizing or decreasing V_t, in most cases. As a result, the sensitivity scores for timing recovery are negative values, and the solutions with larger sensitivity scores (i.e., smaller absolute values) should have higher priorities.

Algorithm 5 Procedure of kick move.

Procedure: $KickMove(th_k, X_k)$
Input: slack threshold for kick move th_k, maximum #trials X_k
Output: an updated netlist
1: $SF \leftarrow 1/(\Delta d_i \cdot \#paths_i)$
2: $TimingRecovery(SF, th_k, X_k)$

Kick move. Algorithm 5 illustrates the $KickMove$ procedure. The purpose of $KickMove$ is to perturb the current status so that the optimization does not get stuck at local minima. Thus, for the $KickMove$ procedure, we perform upsizing on cells that have positive slacks but less than th_k, with a sensitivity function that considers only timing (Line 1). The number of cell moves during the $KickMove$ is limited by X_k. The default number of X_k is set to 0.05% of the total number of cells so as not to allow overly aggressive perturbation.

2.1.4 Experimental Setup and Results

Sizer is implemented with C++ and a *Tcl*-socket interface [199] to communicate with golden signoff timers. Sizer supports the standard SPEF/.v formats by using *OpenAccess (OA) 22.43* [193] API. We extend the Liberty parser provided by ISPD-2013 contest to support general Liberty (.lib) files and collect internal power information of cells. The extended Liberty parser is validated with various foundry libraries, i.e., 140nm, 65nm GP and 28nm LP.

Experimental Setup

Testcases. We use five testcases in our experiments, each of which is implemented with two foundry technologies – 65nm GP with triple-V_t libraries and 28nm LP with dual-V_t libraries. To validate solution quality as well as scalability of Sizer, we use one design (AES) from the *OpenCores* website [191], a simplified version of ARM Cortex M0 (M0), and testcases with three and five copies of the M0 (i.e., M0x3, M0x5) as well as matrix multiplier (MMULT) from the ISPD-2013 benchmark suite. To implement M0x3 and M0x5, we connect primary inputs and outputs of M0 with muxes at the RTL level. We synthesize from RTL to gate-level netlist using *Synopsys Design Compiler H-2013.03-SP3* [195] for AES, M0, M0x3 and M0x5. For MMULT, we map the gate-level netlist from ISPD testcases to given technologies using

Synopsys Design Compiler H-2013.03-SP3 [195]. To perform placement and routing (P&R), we use *Cadence Encounter Digital Implementation System 14.2* [171]. We further optimize the designs using a leading commercial tool, namely, *Comm1*, with high-effort leakage and dynamic power optimizations. For golden signoff timers, we use *Synopsys PrimeTime J-2014.12* [198] (PT) and *Cadence Tempus 14.2* [172] (TMP). The tool versions represent the most up-to-date common flow that could be realized across multiple organizations when this work was performed (see Section 2.1.4). We do not map specific results to specific tools (although we do indicate a "universe" of tools that we have used), in order to avoid any reporting that could possibly be construed to be "benchmarking".

We also validate Sizer on a design from *NXP Semiconductors* [190] implemented with a $140nm$ foundry technology (dual-V_t), which we refer to as *NXPIC*. The design is used in an application of RFCMOS for the keyless entry/go system of automobiles. We apply our Sizer to one of the blocks of NXP's design, which contains $\sim$140K standard-cell instances.

MCMM implementation. To implement the designs in the MCMM context, we define multiple views with various PVT corners and function modes. Table 2.7 summarizes function modes (Column 3), PVT corners (Columns 4 – 6) and wire RC corner (Column 7) for each view in three foundry technologies. We use four views selected from available libraries in $65nm$ and $28nm$ foundry technologies. For $140nm$ foundry technology, we use four representative views (Rows 2–5 of Table 2.7) selected from among 35 views with which the *NXPIC* is implemented.[7] We experimentally confirm that the selected four views are the dominant views among the 35 views based on timing criticality. Ideally, considering all timing views in sizing optimization will produce more robust results that honor timing constraints in every timing view. However, the computation costs increases in proportion to the number of timing views. Thus, the timing views to consider in optimization must be carefully selected to reduce the computational cost. In particular, a timing view having sufficiently large positive slack on its worst timing path will typically not be a dominant view. In the NXPIC case, we exclude the timing views with more than $20\times$ of the worst (positive) slack throughout all the timing views. For example, since the worst initial slack is $0.033ns$ in V1 (Table 2.8), we exclude the timing views with slack values $> 0.66ns$. The timing information in

[7] The 35 timing views are the combinations of three functional modes and three voltages, three process corners (SS, TT and FF) associated with three temperature corners (-40°C, 25°C and 150°C) with complementary views ($\#modes \times \#voltages \times \#corners + \#complementary\ views = 3 \times 3 \times 3 + 8 = 35$).

Table 2.8 is reported by TMP. We do not include TT and FF corners as our optimization corners since (i) setup time violations are dominant in the testcases in our experiments; (ii) downsizing/increasing V_t usually creates setup time violations, and slow corners are more critical during our optimization; and (iii) as we do not touch cells on clock trees, extra (hold) violations are not created at non-slow corners, i.e., TT and FF corners. We experimentally confirm that no extra hold violations are created in optimized designs. In particular, the optimized NXPIC in Table 2.15 does not have any hold or setup violations in the 31 views other than the selected four views. The detailed information of the testcases is summarized in Table 2.8. For $65nm$ and $28nm$ designs, we set the clock period of each view to a view-specific value in order to avoid the situation that a particular view dominates others. Thus, the difference in initial slack values between views is less than or equal to $200ps$. The timing information in Table 2.8 is reported by TMP. For hold slack, we report the worst slack and the sum of total negative slack for the four target views.

Table 2.7: Timing view definitions.

Lib.	View	Mode	Proc.	Volt. (V)	Temp. (C)	RC
$140nm$	V1	Func	SS	1.1	150	Cmax
	V2	Test	SS	1.1	150	Cmax
	V3	Func	SS	1.1	-40	Cmax
	V4	Test	SS	1.1	-40	Cmax
$65nm$	V1	Func	SS	0.9	125	Cmax
	V2	Func	TT	1.0	25	Cmax
	V3	Func	FF	1.1	125	Cmax
	V4	Func	FF	1.1	-40	Cmax
$28nm$	V1	Func	SS	0.9	125	Cmax
	V2	Func	SS	0.9	-40	Cmax
	V3	Func	SS	1.1	125	Cmax
	V4	Func	SS	1.1	-40	Cmax

Design of Experiments. We have conducted four experiments to demonstrate the performance of Sizer as follows.

- **Expt1** shows the comparison between Sizer, a reproduced version of [84] and a commercial leakage optimization tool, namely, *C2*.
- **Expt2** shows our optimization results with MCMM on the optimized designs with high effort option using C1.

Table 2.8: Summary of testcases. The designs are optimized with high-effort optimization options for leakage and dynamic power using C1. The values are reported by TMP.

Lib.	Design	#Cells	Clock Period (ns)				Worst Setup Slack (ns)				Hold Slack (ns)		Power (mW)	
			V1	V2	V3	V4	V1	V2	V3	V4	WNS	TNS	Leakage	Total
$140nm$	NXPIC	140K	NA	NA	NA	NA	0.033	0.104	0.264	0.574	NA	NA	4.17e-5	NA
$65nm$	M0	13139	1.60	1.10	0.80	0.80	-0.003	0.066	0.02	0.046	-0.294	-39	0.40	20.14
	AES	20583	0.90	0.60	0.40	0.40	-0.014	0.032	-0.005	0.013	-0.144	-82	0.78	112.50
	M0x3	35473	1.85	1.15	0.85	0.85	0.001	0.003	0.003	0.035	-0.347	-54	0.86	50.09
	M0x5	55160	1.90	1.20	0.90	0.90	-0.012	-0.017	-0.005	0.03	-0.185	-54	1.23	74.87
	MMULT	123283	2.10	1.40	1.00	1.00	0.002	0.05	0.011	0.043	-0.008	0	3.47	191.98
$28nm$	M0	9964	1.40	1.60	1.00	1.00	0.016	-0.010	0.036	0.013	-0.276	-44	0.16	15.27
	AES	13154	0.90	1.10	0.60	0.60	0.010	0.023	0.011	-0.005	-0.113	-57	0.25	56.76
	M0x3	30155	1.40	1.70	1.00	1.00	-0.273	-0.313	-0.139	-0.183	-0.417	-165	0.79	80.13
	M0x5	50231	1.50	1.85	1.25	1.25	-0.013	-0.017	0.184	0.153	-0.303	-100	0.47	58.77
	MMULT	107619	1.80	2.10	1.30	1.30	-0.011	-0.017	0.037	0.003	0	0	1.66	186.18

- **Expt3** studies the impact of six SFs on the solutions.
- **Expt4** shows the optimization results of a commercial product design in $140nm$.

Table 2.9 summarizes the libraries, tools, signoff tools, the objectives of optimization, whether MCMM is considered, and the sensitivity functions used for each experiment.

Table 2.9: Design of experiments.

	Tech.	Tool	Signoff	Objective	MCMM	SF
Expt1	$65nm$	Tri-R *C2*	PT	Leakage	No	SF3
Expt2	$28nm$ $65nm$	C1	TMP	Leakage Total power	Yes	MSF1
Expt3	$65nm$	–	TMP	Leakage	No	SF1-6
Expt4	$140nm$	C1	TMP	Leakage	Yes	MSF1-2

Expt1: Comparison between Sizer, [84] and a Commercial Tool (*C2*)

In Expt1, we compare the performance of our Sizer against our reproduced version of Trident [84] (*Tri-R*) and C2 on contest benchmarks and five testcases implemented at $65nm$ foundry technology.

Tri-R, reproduced Trident. As Trident [84] is designed to perform for contest testcases, it does not support inputs with standard formats. Thus, we extend Trident to enable support for the standard SPEF/.v/.lib formats. We refer to this extended version of [84] as Tri-R. We confirm that Tri-R reproduces similar results to that in [84] (Table 2.10). The golden signoff timer is PT to be consistent with the golden

Table 2.10: Reproduction of ISPD Trident (Tri) results with Tri-R, and comparison with Sizer (Szr). The values are reported by PT.

Design	#Cells	Leakage (mW)				Runtime (min)		
		Init	Tri	Tri-R	Szr	Tri	Tri-R	Szr
usb_phy_fast	608	1.73	1.59	1.60	1.59	0.21	0.99	2.7
usb_phy_slow	608	1.1	1.07	1.07	1.05	0.17	0.64	1.8
pci_bridge32_fast	30603	145.6	101.90	99.82	113.24	12	23.62	68
pci_bridge32_slow	30603	65.3	58.83	59.04	59.26	5.39	11.95	65
fft_fast	32766	583.04	305.29	305.38	357.01	32.58	116.58	116
fft_slow	32766	128.62	93.10	92.94	99.82	17.4	42.03	69
edit_dist_fast	126665	1040	619.30	613.01	1040	170.6	649.19	1321
edit_dist_slow	126665	63	465.60	464.82	544.09	107.2	337.56	1211

timer used in the ISPD contest. The slight difference in the reported leakage results is due to sensitivity to the order of calculation as well as randomness seen in multi-threaded operation. The longer runtime in Tri-R is due to (i) more complex data structures and extra input processing steps (e.g., reading input SPEF/.v into OA database); and (ii) avoidance of certain hard-wiring of codes used by [84] for speedup. One caveat of academic sizers is that in many cases, hard-wired codes are used to achieve high quality of solutions with minimum runtime for a particular set of inputs (i.e., contest testcases). Un-hardwiring leads to runtime increases. Based on the results, below with the assumption that Trident and Tri-R are equivalent, we compare Tri-R and Sizer with designs at 65nm technology.

We further compare Trident, Tri-R with Sizer. As Sizer is designed to maintain initial timing slack, Sizer cannot handle the contest benchmarks as is, in which every gate is sized to the largest size / lowest V_t and WNS is a huge negative value. Thus, we use intermediate, timing feasible solutions from Trident as the inputs for this experiment. We observe that Sizer produces similar solution qualities for usb_phy_fast, usb_phy_slow and pci_bridge32_slow, but does not perform well for the other contest benchmarks with longer runtimes. This may reveal the gap between academic sizers that are optimized toward academic benchmarks and commercial sizers, as also noted in the studies in [87].

Comparison of Sizer, Tri-R and C2 with 65nm designs. Table 2.11 shows leakage results comparison between Sizer (Szr, Szr-I), Tri-R, and C2. Clock period (CP), worst negative setup slack (WNS), leakage results (Leak) and runtime are reported. The initial/optimized results do not have any hold time violation. Sizer achieves leakage reductions of up to 26% (19% on average) as compared to Tri-R.

Sizer's runtime is longer than that of Tri-R, due to more iterations of optimization. Thus, we also report the intermediate results of Sizer (Szr-I) so as to enable an iso-runtime comparison. For the iso-runtime comparison, our intermediate results also outperform Tri-R by up to 17% (12% on average).

We further compare our results with the results from C2. Sizer achieves further leakage reduction beyond C2 results by up to 24% (19% on average). Although the runtime of C2 is much better (the runtime is less than a minute) than Sizer, these data show that C2 leaves significant room for further leakage optimization. We further observe that we can pay additional runtime for significant additional power optimization using Sizer.

Table 2.11: Leakage optimization result comparison between Tri-R, Sizer (Szr) and a commercial tool (C2). The results are reported by PT.

Design	CP (ns)	WNS (ns)	Leak (mW)	WNS (ns)				ΔLeak				Runtime (min)			
				Tri-R	Szr	Szr-I	C2	Tri-R	Szr	Szr-I	C2	Tri-R	Szr	Szr-I	C2
M0	1.64	0.031	0.40	0.031	0	0.000	0	0%	23%	16%	4%	52	263	44	0
AES	1.0	0.001	0.78	0.001	0	0.000	0	27%	46%	32%	22%	42	200	44	0
M0x3	2.11	0.032	0.87	0.006	0	-0.001	0	32%	54%	47%	30%	99	541	106	0
M0x5	2.16	0.003	1.23	0.003	0	-0.002	0	24%	50%	41%	25%	115	942	126	0
MMULT	2.15	0.034	3.49	0.034	0	0.000	0	0%	6%	6%	2%	414	1410	391	0

Expt2: Sizer Optimization Results with MCMM on the Solutions of C1

In Expt2, we show experimental results with MCMM-aware optimization using Sizer, on designs that have been optimized by C1. To ensure a fair comparison between C1 versus Sizer, we perform an iso-TNS and iso-WNS comparison (with respect to setup time constraints), such that our optimization honors the initial total negative slack and worst negative slack of the design, and does not permit further timing slack degradation. We perform the experiments at both $65nm$ and $28nm$ technologies, with two objectives – leakage power reduction and total power optimizations. The leakage and total power values at V1 view are reported by TMP in Table 2.12. For all designs, there was no extra hold violation after optimization. The initial timing and power information is reported in Table 2.8. Sizer is able to achieve up to 10% leakage reduction and 4% total power reduction on initial solutions implemented with high-effort optimization for both leakage and total power reduction using C1, for $65nm$ designs. For $28nm$ designs, Sizer achieves up to 2% leakage reduction and 4% total power reduction, respectively. Our Sizer could not optimize power further for some designs (e.g., no leakage reduction for $65nm$ M0x5, etc.). Of course,

the initial designs are already optimized by C1, and hence, there is not much timing slack for power optimization to exploit. Indeed, some designs even have negative initial slack; this implies that C1 used up all the slacks, and ended up with timing-infeasible solutions.

However, we believe that any improvement that Sizer achieves indicates that there is still room for further optimization of existing leading commercial sizers. And, although numbers might seem to be small, even 1 or 2% of power reduction could be helpful for designs that have tight power constraints [72]. Even if Sizer's performance is "only" very similar to that of leading-edge industry tools, Sizer provides an important new research platform for academic research, as it gives an industry-strength implementation accompanied by an open, full description. We feel that this is a strong contrast to industry tools, details of which are often kept highly confidential by EDA companies. Sizer source codes and scripts are available in [203].

The larger runtime of multi-corner optimization is mainly due to the timing updates and interface with golden signoff timer (as shown in Column 14, in Table 2.12).

Table 2.12: Leakage and total power optimization results.

Process	Optimization	Design	Setup Slack (ns)				Hold Slack (ns)		Final Power (mW)		ΔPower		Memory	Runtime (min)	
			V1	V2	V3	V4	WNS	TNS	Leak	Total	Leak	Total	(MB)	Total	Timer
65nm	leakage	M0	0.000	0.052	0.019	0.044	-0.294	39	0.37	20.00	**6%**	1%	1680	345	279
		AES	-0.004	0.034	0.000	0.011	-0.144	81	0.70	111.33	**10%**	-3%	2148	1068	944
		M0x3	0.000	-0.001	0.003	0.033	-0.347	53	0.77	49.75	**10%**	1%	3981	2161	1836
		M0x5	-0.012	-0.017	-0.006	0.030	-0.185	54	1.23	74.87	**0%**	0%	6518	1983	1724
		MMULT	0.000	0.050	0.011	0.043	-0.008	0	3.44	191.83	**1%**	0%	10673	498	397
	total	M0	0.000	0.062	0.021	0.051	-0.294	39	0.39	19.66	3%	**2%**	1680	559	466
		AES	0.000	0.032	0.000	0.000	-0.144	81	0.78	108.02	0%	**4%**	2148	1107	972
		M0x3	0.000	0.000	0.004	0.037	-0.347	53	0.80	48.28	7%	**4%**	3981	2293	1956
		M0x5	-0.012	-0.017	-0.006	0.030	-0.185	54	1.23	74.87	0%	**0%**	6518	2406	2075
		MMULT	0.000	0.046	0.006	0.038	-0.009	0	3.52	190.66	-1%	**1%**	10673	3960	3339
28nm	leakage	M0	0.011	-0.012	0.033	0.008	-0.276	43	0.16	15.24	**2%**	0%	1310	75	48
		AES	0.008	0.02	0.01	-0.006	-0.113	57	0.25	56.76	**0%**	0%	1775	224	175
		M0x3	-0.274	-0.317	-0.14	-0.185	-0.417	165	0.47	58.77	**0%**	0%	3928	237	174
		M0x5	-0.03	-0.029	0.168	0.135	-0.305	112	0.78	80.14	**2%**	0%	6355	521	347
		MMULT	-0.011	-0.017	0.037	0.003	0	0	1.66	186.18	**0%**	0%	10658	2226	1144
	total	M0	0.005	-0.011	0.03	0.005	-0.27	36	0.17	14.77	-3%	**3%**	1310	107	68
		AES	0.007	0.026	0.001	-0.022	-0.113	43	0.26	56.53	-6%	**0%**	1774	383	316
		M0x3	-0.261	-0.364	-0.136	-0.178	-0.417	91	0.50	57.41	-8%	**2%**	3941	406	234
		M0x5	-0.018	-0.027	0.184	0.153	-0.305	82	0.81	76.61	-2%	**4%**	6355	598	349
		MMULT	-0.014	-0.021	0.035	0	0	0	1.66	186.25	0%	**0%**	10658	2551	1276

Expt3: Impact of Various SFs on the Solutions

Expt3 studies power reductions and runtime of various sensitivity functions. Table 2.13 shows that sensitivity function SF3 leads to the best solution quality, and that sensitivity function is a key determinant

of solution quality and runtime. In general, we observe that the $\#paths$ parameter does not help to achieve better results for the testcases in our experiments. For example, SF2 and SF4 show the impact of $\#paths$; SF2 (without $\#paths$) leads to better results than SF4 (with $\#paths$). SFs without the parameter $\#paths$ (i.e., SF1, SF2 and SF3) offer better results than those with the parameter $\#paths$ (i.e., SF4, SF5 and SF6). Table 2.14 shows the leakage and runtime results for the three MSFs. MSF3 does not give any power reduction since MSF3 is also used for timing recovery. Indeed, if the same SF is used for power reduction and timing recovery, it is likely that the same set of gates will be selected for both power reduction and timing recovery. Between MSF1 and MSF2, we see that MSF2 has better performance in terms of power reduction.

Table 2.13: Leakage optimization results of $65nm$ designs with various SF.

Design	**ΔLeakage**						**Runtime (min)**					
	SF1	**SF2**	**SF3**	**SF4**	**SF5**	**SF6**	**SF1**	**SF2**	**SF3**	**SF4**	**SF5**	**SF6**
M0	14%	16%	23%	1%	0%	3%	256	143	263	143	99	305
AES	26%	31%	46%	8%	8%	13%	242	149	200	163	103	271
M0x3	45%	51%	54%	4%	10%	16%	404	359	541	124	200	873
M0x5	41%	47%	50%	6%	10%	12%	680	1001	942	387	359	1036
MMULT	4%	4%	6%	0%	0%	0%	1838	1470	1410	1568	1171	1330

Expt4: Results of Commercial Industrial Design in 140nm

In Expt4, we apply Sizer to an industrial design in $140nm$ technology. Our results in Table 2.15 show that Sizer can achieve 7.1% leakage power reduction beyond an input solution that is well-optimized by commercial P&R tools and ECO optimizations. In this experiment, the integrated use of the signoff timer has large runtime overhead due to the following reasons. First, the signoff is signal integrity (SI)-

Table 2.14: Leakage optimization results of $65nm$ designs with various MSF.

Design	**ΔLeakage**			**Runtime (min)**		
	MSF1	**MSF2**	**MSF3**	**MSF1**	**MSF2**	**MSF3**
M0	7%	6%	0%	62	345	48
AES	6%	10%	0%	216	1068	125
M0x3	4%	10%	0%	360	2161	508
M0x5	0%	0%	0%	313	1983	536
MMULT	3%	1%	0%	477	498	384

aware; to perform SI-aware timing analysis, large databases of timing noise information are loaded and the full timing analysis takes several minutes. Second, in our experiments, accurate incremental timing analysis is not available due to a tool limitation. For these reasons, 90% of the total runtime on average is consumed by the signoff timer. Due to the runtime issue of the signoff timer, we run Sizer several times with a limited number of cell swaps (i.e., 1000) and a limited number of optimizations (i.e., one). We report the number of cell swaps (second column) and leakage power (eighth column) for each iteration. One iteration consists of both the power reduction and timing recovery steps. After the ninth iteration, we observe that the leakage does not reduce further with MSF2.[8] Changing from MSF2 to MSF1 offers an additional 1.3% reduction.

Table 2.15: Leakage optimization results for NXPIC.

Iter	**#Swaps**	**SF**	**Final WNS (ns)**				**ΔLeakage**	**Runtime (min)**
			V1	**V2**	**V3**	**V4**		
1	181	MSF2	0.033	0.104	0.264	0.574	1.44%	417
2	889	MSF2	0.033	0.104	0.264	0.574	4.02%	1045
3	117	MSF2	0.033	0.104	0.032	0.034	4.31%	437
4	120	MSF2	0.033	0.104	0.032	0.034	4.57%	420
5	36	MSF2	0.033	0.104	0.032	0.034	4.67%	330
6	12	MSF2	0.033	0.104	0.032	0.034	4.70%	362
7	155	MSF2	0.033	0.104	0.032	0.034	4.85%	507
8	40	MSF2	0.033	0.104	0.032	0.034	4.96%	395
9	222	MSF2	0.033	0.104	0.032	0.034	5.29%	856
10	201	MSF1	0.033	0.104	0.032	0.034	6.99%	874
11	28	MSF1	0.033	0.104	0.101	0.060	7.10%	480

2.1.5 Conclusion

Gate sizing optimization has been well studied, and academic sizers have shown significant power reduction on contest testcases [119] [120]. However, in the real world, design complications (hierarchical design, existence of hard macros, multiple clocks), modeling complications (state-dependence, complex slew dependence), timing constraints (MCMM, false paths), and electrical constraints (MaxTran, MaxCap) block the application of academic sizers to industrial designs. More crucially, the real world can reveal that

[8] Several additional iterations were attempted with MSF2, but no improvement was observed.

techniques and "best practices" identified by academic research (driven by popular contest benchmarks) may not be usable on industrial designs. Our new academic sizer, *Sizer*, reflects a multi-year evolution from a successful "contest" sizing tool to a tool that can outperform high-effort commercial results for a real industrial IC. This evolution has entailed a near-complete change of techniques as compared to our starting point, *Trident* [84]. We describe techniques that are useful in the academic contest setting but not in the real-world context – as well as new techniques specifically developed for Sizer. We also describe the successful application of Sizer to industrial designs. On a design (i.e., *NXPIC*) that is well-optimized by a leading-edge commercial P&R tool as well as ECO steps, Sizer achieves 7% further leakage reduction without any violation of the given setup/hold and maximum transition constraints in a multi-corner multi-mode context.

Our future works include (i) runtime improvement with better timing recovery; (ii) improved, stochastic optimization by introducing different SFs in $KickMove$ function and a random parameter; and (iii) considerations of other important constraints in industry designs such as noise.

2.2 Minimum Implant Area-Aware Gate Sizing and Placement

As minimum feature sizes decrease, physical design rules have become tighter. Geometric constraints for layout that arise from limits of patterning technology are described as design rules in technology files such as LEF [186]. Each layer has width, spacing, minimum-area, etc. rules which can affect the legality of standard-cell placement. Implant (active) layers, which indicate regions for ion implantation, determine the threshold voltage (V_t) of transistors. Figure 2.5 shows the implant region in standard cells and an example *minimum implant area* (MinIA) layer rule in the LEF file. As shown in the figure, the polygons of implant layers have drawn dimensions typically matched to the width of standard cells.

MinIA layer rules affect multi-V_t standard cell-based designs; the multi-V_t regime is essential to achieve low-leakage, high-performance design implementations [184]. Traditional timing- and routability-driven placement of cells with multiple V_t values can result in a small island of a given V_t that cannot meet the MinIA layer rule. A small cell that cannot meet the MinIA rule must be abutted with cells having

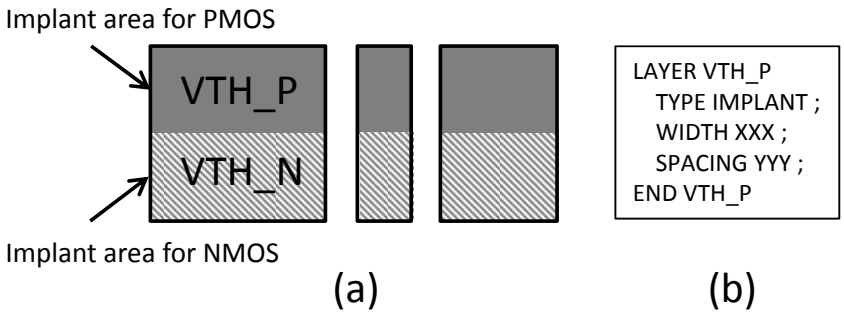


Figure 2.5: (a) Illustration of implant layer regions in standard cells. (b) An example of an implant layer rule in a LEF5.7 file.

the same V_t, so as to form a wider implant layer polygon. That is, a narrow cell cannot be sandwiched between different-V_t cells. Figure 2.6 shows an example of a MinIA violation. The dotted line indicates the minimum width of the V_{t_2} implant layer. A narrow cell c_2 with V_{t_2} is surrounded by V_{t_1} cells, and this violates the MinIA (equivalently, width) constraint. We note that the impact of the MinIA rule can be huge when the proportion of small-width cells in the netlist is large; this is a common scenario especially in cost-driven, low-power mobile IC products. We have studied how "thin" a netlist can be. For example, in a *jpeg* netlist synthesized from *OpenCores* [191] RTL using a $28nm$ FDSOI foundry library, the smallest and second-smallest cell widths comprise $\sim$18% and $\sim$23% of the total number of instances, respectively.

The (minimum width and spacing) design rules for implant layers have not been critical before, as cell sizes have been large enough to cover these minimum rules. Hence, up until recent technology generations, placement and gate sizing/V_t-swapping methods have not had to consider these rules, since any legal cell placement would be correct with respect to the MinIA criterion. However, as cell sizes have continued to shrink in advanced process nodes, even as the wavelength used in $193i$ optical lithography remains constant, the MinIA rule has become larger than the minimum width of standard cells (e.g., INV$\times$1 cell). MinIA rules constrain cell placement starting with the foundry "$20nm$" ($64nm$ minimum metal pitch) node, due to the minimum pattern size limits and cell library (diffusion layer layout) strategies.

The minimum width constraint of implant layers changes the traditional placement and post-layout gate sizing problems. That is, beyond existing constraints such as timing, power and area, additional *geometric* information of cells must be considered. A major change to the traditional sizing problem formulation is that even downsizing and V_t-swapping operations must comprehend spatially adjacent

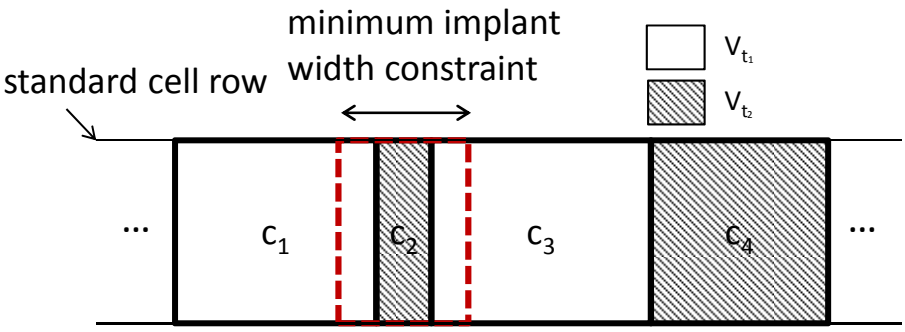


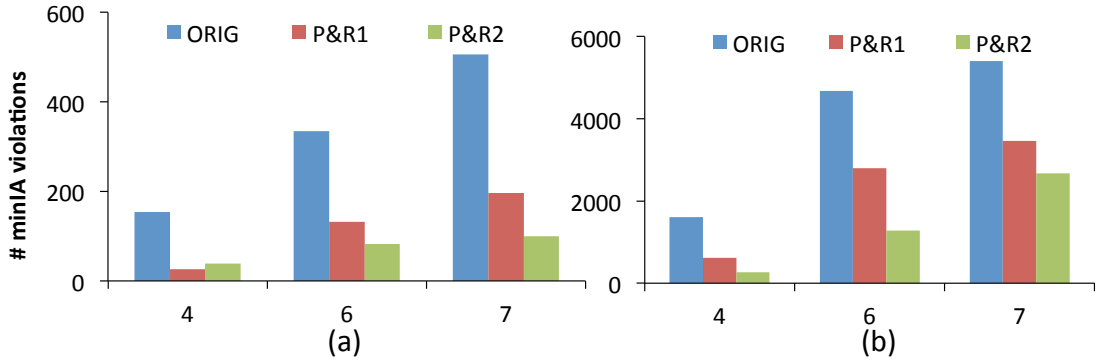
Figure 2.6: An example of the minimum implant area violation. The dotted line indicates the minimum width constraint of the implant layer. The cell instance c_2 (V_{t_2}) violates the constraint as it is narrow and sandwiched by two cells (c_1 and c_3) that have a different V_t (V_{t_1}).

cell instances and whitespace in order to avoid creation of sandwiched narrow cells. Further, placement algorithms must comprehend V_t assignment of spatially adjacent cell instances, as well as whitespace, for the same reason. Therefore, the two problems cannot be solved independently. A unified method that considers both cell size/V_t and placement together is needed.

Case Study of P&R Tools

Recently, commercial P&R tools have claimed that minimum implant area rules are considered during the implementation, given that filler cells with active layers can be used to improve active density and manage STI stress effects [135]. Commercial P&R tools apparently fix the minimum implant area violations by inserting filler cells at the final design stage. For example, Mentor Graphics *Olympus* [189] has a utility to define an implant layer group for filler cells, so that each narrow cell can be padded filler cells having the same implant type. Cadence *Encounter Digital Implementation System (EDI)* [171] checks and fixes implant area violations according to the rules specified in LEF, during placement and filler cell insertion. Synopsys *IC Compiler (ICC)* [197] offers a *Voltage-threshold-aware filler cell insertion* flow according to which users can define the V_t filler cells to be inserted between different V_t regions. For example, users can insert NVT filler cells between NVT and HVT cells, and LVT filler cells between LVT and NVT cells. In our case studies of P&R tools, we assign HVT filler cells to HVT-NVT and HVT-LVT pairs, and NVT filler cells to NVT-LVT pairs. We have tested commercial P&R tools using two small netlists (DMA and AES) synthesized from *OpenCores* [191] RTL, and a 45nm foundry library along

with intentionally tightened MinIA constraints. The minimum implant layer rules are defined in LEF. The portion of each V_t type in the testcases is evenly distributed so as to heighten the number of MinIA rule violations. The utilizations in the DMA and AES implementations are 77% and 82%, respectively. Figure 2.7(c) shows example commands used with current P&R tools. After routing, the tools' built-in filler cell insertion flows are applied. The results shown in Figures 2.7 (a) and (b) show that commercial P&R tools cannot fix all of the MinIA violations by simply inserting filler cells.



P&R tool script examples	
verifyGeometry addFiller {cell_list}	set_cell_vt_type set_vt_filler_rule "VT1 VT2" {cell_list} insert_stdcell_filler

(c)

Figure 2.7: Minimum implant area violation fix results from two commercial tools (P&R1 and P&R2) through filler cell insertion, for two testcases: (a) DMA and (b) AES. The x-axis shows the minimum implant width constraints in the number of grids. (c) The commands used for filler cell insertion flow.

This Work

We have discussed how MinIA rules change the gate sizing and placement problems, and we have observed that commercial P&R tools can handle this new issue only to a limited extent. To our knowledge, no work in the research literature has tried to solve MinIA violations systematically, for both gate sizing and placement. The main contributions of our work are summarized as follows.

- We redefine traditional placement and gate sizing problems in view of the minimum implant layer constraint.

- We propose methods to minimize MinIA violations and optimize power under the MinIA constraint with placement perturbation and gate sizing/V_t-swapping.

- Our proposed methods are implemented with C++ programs and incorporated into a standard P&R flow. Our methods are validated with a commercial tool and $45nm$ foundry library with a range of MinIA constraints.

2.2.1 Related Work

Gate sizing, and co-optimization with placement. Traditional gate sizing, which optimizes power and delay of circuits using size (gate width), V_t, and gate length of cells, has been extensively reviewed in [98]. Co-optimization of placement and gate sizing has had limited previous consideration. Lee and Gupta [97] suggest LP-based gate sizing considering an ECO cost, which is computed with respect to power and area and modeled as a linear function of several parameters. The objective is to minimize "power + ECO cost". Luo et al. [114] minimize power by combining placement, sizing and V_t-swapping optimizations using a slack management technique. The placement, sizing and V_t-swapping phases are performed sequentially.

Linear (1-D) placement. Since MinIA violations do not occur between standard cell rows, the associated problem of obtaining a MinIA-legal placement can be treated as a type of linear (1-D) placement. In the related literature, Kahng et al. [92] and Brenner et al. [16] consider the problem of placing a set of cells in a single row with a fixed horizontal ordering, with the objective of minimizing the (weighted) sum of net bounding box perimeters. The paper [89] evaluates several techniques to legalize cell overlaps in row-based placements, so as to improve metrics of routability and routed wirelength. In [56], Gupta et al. describe a dynamic programming (DP) based technique for Assist-Feature Correctness (AFCorr) in detailed placement of standard-cell designs. Their implementation achieves improved depth of focus and CD control, and subsequent works from the same group use similar DP methods to address leakage and other objectives.

Layout effect-aware placement. STI stress-aware placement has tangential similarity to our MinIA placement, in the sense that solutions can seek to increase implant area; however, the main objective

is different. Joshi et al. [80] propose stress as a means to achieve an optimal power-performance tradeoff. They study how stress-induced performance enhancements are affected by layout properties, and they improve standard-cell layouts to achieve maximum performance gain with dual-V_t assignment. Kahng et al. [91] combine detailed placement and active-layer fill insertion to exploit STI stress for performance improvement. Chakraborty et al. [21] propose an active area sizing aware cell-level delay model which forms the basis of linear programming to achieve either maximum performance, or target performance under a resource budget. Li et al. [102] present a methodology to determine optimal STI well width; using geometric programming, they also solve a STI stress aware placement optimization formulation.

2.2.2 Problem Formulation

Given a placed standard-cell design[9] with potential MinIA rule violations, our ultimate goal is to find location and sizing/V_t assignment for each cell, so as to achieve minimum power without any violation of design constraints – including timing, placement legality, and MinIA rule constraints. Implicitly, our problem formulation assumes the following. (1) We do not consider violations that occur inside of a cell; individual cells are assumed correct by construction. (2) We assume that the minimum width of an implant region inside a cell is always the same as the cell width. (3) We do not consider the height of implant regions within a cell; again, correctness by construction is assumed. To optimize power considering both timing and geometry information, we combine gate sizing and ECO placement to address MinIA-aware gate sizing and placement problem:

[9] Possibly, the design is routed as well, depending on preferences regarding parasitic estimation accuracy versus turnaround time.

Problem: $MinIA$**-aware sizing and placement**

Minimize: $\sum_{\forall c} P(c)$

Subject to:

$$S(c) > 0 \quad \text{(T1)}$$

$$T_r(c) < T_{r_{max}} \quad \text{(T2)}$$

$$W_e(c) < I_{min}(V_t(c)), \ \ \forall c \quad \text{(P1)}$$

$$\text{No overlap in placement} \quad \text{(P2)}$$

Here, (T1) and (T2) are the timing-related constraints: (T1) is the setup (max path delay) slack constraint, and (T2) is the maximum transition time constraint. (P1) and (P2) are the placement-related constraints: (P1) is the minimum implant area constraint, and (P2) is the placement legality (non-overlapping) constraint. Additional constraints such as max capacitance, hold time checks, and various other physical design rules can be transparently considered within the approaches that we propose. We omit discussion of these for simplicity of exposition.

Table 2.16: Notations used in this work.

Notation	Meaning
$P(c)$	leakage power of cell c
$V_t(c)$	threshold voltage of cell c
$S(c)$	timing slack of cell c
$T_r(c)$	transition time of cell c
$T_{r_{max}}$	max transition time constraint
$W_e(c)$	effective implant layer width of cell c[10]
$B(c)$	violations caused by cell c in placement[11]
$\{N(c)\}$	neighbor cells of cell c
$I_{min}(V_t)$	minimum implant area constraint for V_t
m_k	a potential sizing solution

When we optimize both sizing and ECO placement via any sequential/iterative procedure, it is clear that gate sizing will minimize power at the cost of potential violations (in particular, with respect to

[10] The maximum width of any contiguous region of an implant layer in c.

[11] $B(c)$ includes minimum implant area violations.

MinIA placement rules), and that these violations must be fixed by an ECO placement step. Further, the sizing and placement problems naturally have different objectives and constraints. How we combine the two problems in a single framework will induce any of several basic heuristic approaches:

Heur1. Size cells freely (enforcing only the (T1)(T2) constraints, as is traditional in gate-sizing formulations), and fix placement at a later stage (enforcing all of (T1)(T2)(P1)(P2)).

Heur2. Constrain sizing to enforce all placement and design rules (i.e., with enforcement of (T1)(T2)(P1)(P2)).

Heur3. Size cells with partial constraints of placement (enforcing (T1)(T2) constraints, and relaxed (P1)(P2) constraints), such that only a small number of violations require fixing at the ECO placement stage.

We observe that **Heur1** may achieve the best power reduction with its sizing optimization, but may also result in a large number of MinIA violations in placement. At the other extreme, **Heur2** may achieve only limited power reduction since all potential sizing moves for each cell are restricted by placement. **Heur3** may be viewed as a compromise between the two methods. We note that **Heur3** gives the best results in our experiments reported in Section 2.2.4 below.

A truly simultaneous optimization of sizing and placement for MinIA fixing is not obvious to us at this point: it appears difficult to explore the entire solution space since there are so many combinations of sizes, locations, V_t assignments, and filler cell assignments. For now, we have pursued sequential optimization of sizing and placement, with (i) a sizing heuristic that considers placement, and (ii) placement perturbation heuristics to fix post-sizing MinIA violations. For the placement optimization stage, we define a $MinIA$**-aware placement** problem, derived from the $MinIA$**-aware sizing and placement** problem,

in which the objective is to minimize the number of violations:

$$\textbf{Problem:} \quad MinIA\textbf{-aware placement}$$

$$\textbf{Minimize:} \quad \sum_{\forall c} B(c)$$

$$\textbf{Subject to:} \quad S_c > 0 \tag{T1}$$

$$T_r(c) < T_{r_{max}} \tag{T2}$$

The **Experiment 1** discussion in Section 2.2.4 below assesses our placement algorithms in the context of this $MinIA$**-aware placement** problem.

2.2.3 Our Approach

We now discuss our gate sizing and placement approaches considering MinIA constraints.

Minimum Implant Area-Aware Placement

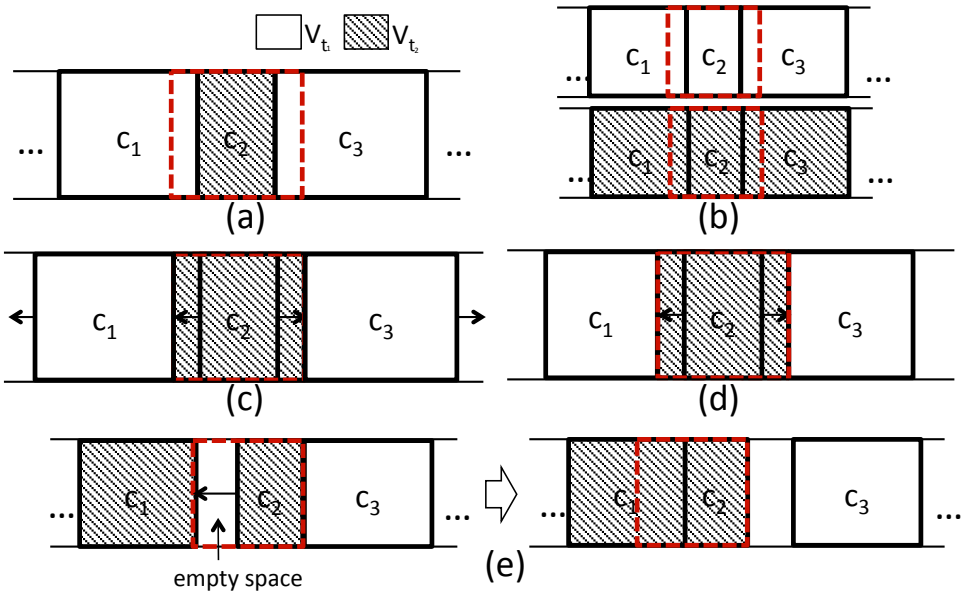


Figure 2.8: Available fixing approaches for MinIA rule violations. A given violation, depicted in (a), can be fixed by using (b) V_t-swapping, or (c) moving a neighbor cell, or (d) downsizing a neighbor, or (e) moving the narrow cell.

Levers to solve MinIA rule violation. If violators of MinIA constraints have enough spacing around them, then the violations can be easily fixed by inserting same-V_t filler cells, thereby increasing the width of the implant area. However, when the spacing is insufficient, we must change cell size/V_t or perturb the placement, such that the implant layer region of the narrow cell can match up with an adjacent (abutting) implant layer region. Figure 2.8 illustrates four ways to fix a given violation. Suppose that cell c_2 with V_{t_2} violates the MinIA constraint, and that c_1 and c_3 are neighbor cells with V_{t_1}, as shown in Figure 2.8(a). First, we can fix the violation either by swapping V_t of c_2 to V_{t_1} or by swapping V_t of c_1 and c_3 to V_{t_2}; this is shown in Figure 2.8(b). Second, we can push out (i.e., shift) c_1 and c_3 to create spacing for filler cell insertion, as shown in Figure 2.8(c). Third, c_1 and c_3 can be downsized to create spacing around c_2, as shown in Figure 2.8(d). Fourth, the violator can be relocated so that it becomes abutted to a same-V_t cell, c_1 in Figure 2.8(e).

Algorithm 6 $MinIA$-aware Placement Heuristic

Procedure $FixMinImpVio(c, \{I_{min}\}, D)$
Input : a cell c, a set of min implant constraints $\{I_{min}\}$, a netlist D, a placement of D
Output : a sizing/location solution for $c \in D$

1: **for all** cell $c \in D$ **do**
2: $s \leftarrow 0$;
3: **for all** cell $n \in \{N(c)\}$ **do**
4: $s \leftarrow s +$ spacing of c to n;
5: **end for**
6: $MinImpSlack \leftarrow c.width - I_{min}(V_t(c))$;
7: **if** $s + MinImpSlack \geq 0$ **then** insert filler cells;
8: **end for**
9: **if** #violations is zero **then** return success;
10: **for all** cell $c \in D$ **do**
11: **if** cell c violates **then** $ChangeV_tCell(c)$;
12: **end for**
13: **if** #violations is zero **then** return success;
14: **for all** cell $c \in D$ **do**
15: **if** cell c violates **then** $MoveNCell(c)$;
16: **end for**
17: **if** #violations is zero **then** return success;
18: **for all** cell $c \in D$ **do**
19: **if** cell c violates **then** $DownSizeNCell(c)$;
20: **end for**
21: **if** #violations is zero **then** return success;

Heuristic approach for fixing minimum implant area violations. Algorithm 6 shows the overall flow of our heuristic approach, which is based on the four MinIA violation-fixing approaches noted above. First, whitespace around violating cells is calculated (Line 3). If there is enough spacing, sufficient width of same-V_t filler cells is inserted to fix the violation (Line 8). In the next step, V_t-swapping is performed for the violating cell or its neighbor cells with $ChangeV_tCell$ (Line 13). For any violations remaining after this step, there is no space for insertion of filler cells and V_t-swapping is unavailable due to timing constraints. Thus, we try to create spacing by changing the placement. We first try to move neighbor cells (Line 17).[12] Then, downsizing of neighbor cells (Line 21) can be tried if those cells are not timing-critical. Note that these steps can be performed in a different order – e.g., downsizing of neighbor cells can be performed before moving cells. The particular sequence of optimizations of steps used in our flow has been experimentally determined. Filler insertion is performed first, since it does not require any cost. Our studies of the permutations of *V_t-swapping*, *moving* and *downsizing cells* indicate that downsizing of cells occurs very rarely, since many cells are small and timing violations can result. With respect to V_t-swapping and moving cells, the results are better when V_t-swapping is performed first (e.g. #MinIA violations of *V_t-swapping-first* and *moving-first* are 155 and 44, respectively, for the *jpeg* testcase). We also note that our heuristic flow executes steps occur in an order that minimizes perturbation of placement or sizing.

Details of two levers are described in Subroutine 1. $ChangeV_tCell$ tries V_t-swapping for c, using any V_t in $\{N(c)\}$, and checks timing and MinIA violations. If there is any violation, the V_t-swapping of c is reverted (Line 10). In a similar way, V_t-swapping of neighbor cells is tried. $MoveNCell(c)$ moves neighbor cells to create additional space. It first checks whitespace around the left and right neighbor cells and moves these cells if possible, similar to [91]. We limit the movement of cells to avoid large perturbation of placement, in light of the timing impact of cell placement.[13]

Algorithm 6 can be used standalone, in an ECO methodology, to fix MinIA violations. We evaluate our approach with various minimum implant area constraints in Section 2.2.4, **Experiment 1**.

[12] We do not need to try moving the target cell (Figure 2.8(e)) as this case could have be fixed at the filler cell insertion stage.

[13] We observe that up to 10 placement grids of movement will change timing by less than $2ps$ with the $45nm$ foundry library. We allow perturbation of cell location by up to 10 grids.

Subroutine 1 Functions for $MinIA$-aware Placement Heuristic

Procedure $ChangeV_tCell(c)$
Input : a cell c
Output : V_t solutions for c and its neighbor cells
// Change V_t of the violator
$V_{t_{orig}} \leftarrow V_t(c)$;
for all cell $n \in \{N(c)\}$ **do**
  $V_t(c) \leftarrow V_t(n)$;
  $w \leftarrow$ total width of c and $\{N(c)\}$;
  if $w \geq I_{min}(V_t(c))$ && c does not violate timing **then** return success;
  else $V_t(c) \leftarrow V_{t_{orig}}$;
end for
// V_t-swapping for neighbors
for all cell $n \in \{N(c)\}$ **do**
  $V_{t_{orig}} \leftarrow V_t(n)$; $V_t(n) \leftarrow V_t(c)$;
  if n violates timing **then**
    $V_t(n) \leftarrow V_{t_{orig}}$; continue;
  end if
  $w \leftarrow$ total width of c and its neighbor cells;
  if $w \geq I_{min}(V_t(c))$ **then** return success;
end for

Procedure $MoveNCell(c)$
Input : a cell c
Output : location solutions for c's neighbor cells
$n_{cl} \leftarrow$ the left neighbor cell; $n_{cr} \leftarrow$ the right neighbor cell;
$s \leftarrow$ spacing of c to n_{cl} and n_{cr};
$MinImpSlack \leftarrow c.width - I_{min}(V_t(c))$;
if n_{cl} has leftside space Δ **then**
  $d \leftarrow max(-(MinImpSlack + s), \Delta)$; Move n_{cl} by d; $s \leftarrow s + d$;
end if
if $s + MinImpSlack < 0$ **then**
  if n_{cr} has rightside space Δ **then**
    $d \leftarrow max(-(MinImpSlack + s), \Delta)$; Move n_{cr} by d; $s \leftarrow s + d$;
  end if
end if
if $s + MinImpSlack \geq 0$ **then**
  insert filler cells; return success;
else
  return fail;
end if

Minimum Implant Area-Aware Gate Sizing

Our gate sizing method is based on sensitivity-guided gate sizing [66] [84]. In addition to timing constraints, we consider the placement constraints (P1)(P2) described in Section 2.2.2. Our objective is to minimize power without creating any additional (P1)(P2) violations.

Algorithm 7 $MinIA$-aware Gate Sizing Heuristic

Procedure $GSMinImp(\{I_{min}\}, D)$
Input : minimum implant constraints $\{I_{min}\}$, a netlist D, a placement of D, a set of timing constraints
Output : a sizing solution

$M \leftarrow \emptyset$
for all cell c in the netlist D **do**
  if c is downsizable **then**
    $m_k.c \leftarrow c$; $m_k.m \leftarrow downsize$; $m_k.cost \leftarrow \Delta TNS$;
    $m_k.sensitivity \leftarrow \Delta Leakage / m_k.cost$;
    $M \leftarrow M \cup m_k$;
  end if
  if c_i is not a highest V_t cell **then**
    $m_k.c \leftarrow c$; $m_k.m \leftarrow V_t\ upscaling$; $m_k.cost \leftarrow \Delta TNS$;
    $V_{t_{orig}} \leftarrow V_t(c_i)$;
    $V_t(c) \leftarrow$ higher V_t; // placement cost calculation
    $\{N(c)\} \leftarrow \{N(c)\} \cup c$
    for all $n \in \{N(c)\}$ **do**
      if n violates $I_{min}(V_t(n))$ **then**
        if fixable by V_t-swapping/sizing/move of neighbors n **then**
          $m_k.cost \leftarrow m_k.cost + CalCost(n)$;
        else
          break; continue to the next c_i;
        end if
      end if
    end for
    $V_t(c) \leftarrow V_{t_{orig}}$; $m_k.sensitivity \leftarrow \Delta Leakage / m_k.cost$; $M \leftarrow M \cup m_k$;
  end if
end for
while $M \neq \emptyset$ **do**
  Pick a m_k with maximum $sensitivity$ in M; Commit m_k; $M \leftarrow M \setminus m_k$;
  $STA()$;
  Fix the extra MinIA violations;
  if $!feasible()$ **then** restore m_k;
end while

For each gate, we check whether the potential sizing produces violations that require placement

perturbations or sizing of neighbor cells to be resolved. If the neighbor cells need to be changed or relocated, we estimate the timing impact on neighbor cells and add this to the sensitivity function. During gate sizing to recover power, a gate can be downsized or its V_t can be swapped to a higher threshold voltage. Downsizing a gate can produce violations, but if neighbor cells do not consume the whitespace created by downsizing, the violations can be easily cured by inserting filler cells. However, when a gate is V_t-swapped and creates violations by itself or in relation to its neighbor cells, possible fixing methods should be carefully explored. Algorithm 7 describes our sizing flow. $\Delta Leakage/cost$ is used as our sensitivity function. The default cost is the change in total negative slack (TNS). When MinIA violations occur, we additionally calculate potential decrease (worsening) of TNS from changing neighbor cells to fix the violations ($CalCost()$, Line 16). This calculated cost is then added to $cost$ so that the $sensitivity$ decreases.

Overall Flow

Figure 2.9 shows the overall flow of our optimizer, *MinIAOpt*. A DEF file of a routed netlist, and LEF files for geometry information of standard cells and technology information including the minimum implant layer rules, are converted into *OpenAccess* [193] DB using *def/lef2oa* parsers. The minimum implant area-aware gate sizing is performed to reduce leakage power with considerations of geometry information. Further minimum implant area-aware placement optimization can be performed to fix MinIA violations without creating new timing violations. For both stages, a *Tcl*-socket interface is used to enable the communication between the P&R tool and/or timer tool and our optimizer. Via this interface, *MinIAOpt* can send commands to insert filler cells, size/V_t-swap cells and change the locations of cells, as well as obtain updated timing information after ECO routing. We also can use a quick timing estimation from timer tools without ECO routing to achieve faster runtime while sacrificing some accuracy.[14]

[14]For any change of cells, ECO routing should be performed and timing should be updated accordingly. But, performing ECO routing for each cell change takes too much time to be practically feasible. To compensate for the inaccuracy, a timing guardband can be used.

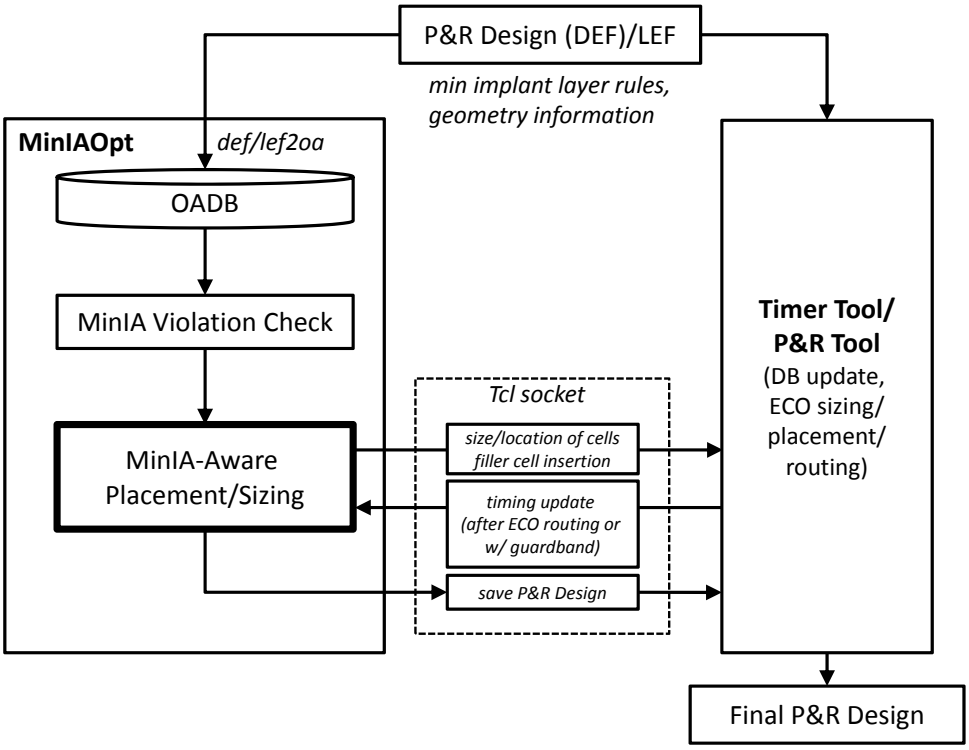


Figure 2.9: Overall flow of our optimizer, *MinIAOpt*.

2.2.4 Experimental Setup and Results

Experimental Setup

Our program is written in C++, and the interface to support DEF/LEF [186] is implemented using the *OpenAccess 2.6* [193] API. We use a *Tcl*-socket interface (*Tcl/Tk 8.4* [199]) to communicate with P&R and timer tools similar to Trident [84] and SensOpt [202]. We have applied our proposed method to a set of open-source designs [191], which we synthesize from RTL using *Synopsys Design Compiler H-2013.03-SP3* [195]. For P&R, we use *Cadence Encounter Digital Implementation System 13.1* [171]. Implementations in all experiments are with a 45nm foundry technology and library.

Table 2.17 shows the testcases used in our experiments. We compare to the result of a simple filler cell insertion performed by a commercial P&R tool with high-utilization testcases that are synthesized with a standard implementation flow. We test various MinIA constraints to understand the scaling of algorithm performance with instance difficulty (e.g., reflecting MinIA constraints in future technology nodes). The

utilization, the distribution of cell V_t values[15], and the percentage of smaller cells (% Mincells) with width less than the minimum implant area constraints all affect the difficulty of a given testcase. We use various minimum implant width constraints $Const1$, $Const2$ and $Const3$, corresponding to four, six and seven, respectively. In the $45nm$ library, 3%, 12% and 28% of standard cells are narrower than these constraints. Also, to study the sensitivity of the results to the difficulty of problem instances, we intentionally tweak the V_t cell distribution of AES and generate *AES_var** so that the same placement will have various numbers of MinIA violations. All experiments are performed on a 2.5GHz Intel Xeon Linux workstation.

Table 2.17: Testcases used in the experiments. *WS, Period, Leak* respectively indicate worst slack, clock period and leakage power after routing, before filler cell insertion.

Bench	**#Inst**	**Util (%)**	**Mincells (%)**	V_t**(H/N/L) (%)**	**WS** (ps)	**Period** (ns)	**Leak** (mW)
DMA	1168	78	59	78/4/16	137	1.0	0.050
MPEG	7121	82	64	83/6/9	39	1.25	0.363
AES	9611	75	84	95/1/2	21	1.5	0.238
JPEG	44911	81	68	82/8/8	36	1.6	2.413
AES_var1	9611	75	84	40/30/30	39	2.1	0.129
AES_var2	9611	75	84	60/20/20	82	2.4	0.106
AES_var3	9611	75	84	80/10/10	137	2.5	0.080

Experimental Results

Table 2.18: Results for a simple filler insertion and our heuristic method.

Bench	**MinIA Const**	**Orig. #Vio.**	**Commercial P&R**	**Heuristic**				
			Δ #Vio. (%)	**Δ WS** (ps)	**Δ Leak** (mW)	**Fill (%)**	**Δ #Vio. (%)**	**CPU total / tool (min)**
DMA	*Const1*	71	-53 (-75)	0	0.000	0.4	-71 (-100)	1.9 / 1.0
	Const2	128	-93 (-73)	0	0.000	1.7	-128 (-100)	2.1 / 1.1
	Const3	193	-151 (-78)	0	0.000	2.5	-193 (-100)	2.0 / 1.1
MPEG	*Const1*	183	-155 (-85)	0	0.000	0.1	-183 (-100)	4.6 / 3.7
	Const2	453	-322 (-71)	-1	0.000	0.7	-451 (-100)	5.9 / 5.0
	Const3	693	-515 (-74)	-18	0.000	1.0	-689 (-99)	6.7 / 5.8
AES	*Const1*	338	-327 (-97)	0	0.000	0.5	-338 (-100)	5.1 / 4.1
	Const2	978	-868 (-89)	0	0.000	2.6	-978 (-100)	8.2 / 7.2
	Const3	1146	-1005 (-88)	0	0.000	3.4	-1146 (-100)	8.7 / 7.8
JPEG	*Const1*	1341	-1186 (-88)	0	0.001	0.3	-1341 (-100)	25.3 / 24.3
	Const2	3865	-3079 (-80)	0	0.004	1.5	-3850 (-100)	73.8 / 72.9
	Const3	7864	-6077 (-77)	0	-0.002	2.5	-7820 (-99)	168.7 / 167.7
AES_var1	*Const3*	2955	-1069 (-36)	28	0.001	10.8	-2863 (-97)	32.3 / 31.4
AES_var2		2558	-1106 (-43)	-51	-0.002	8.5	-2492 (-97)	25.0 / 24.1
AES_var3		1816	-1014 (-56)	-84	-0.002	4.1	-1792 (-99)	15.6 / 14.7

[15]The percentage of each V_t type relative to the total number of cells. H/N/L indicates HVT, NVT and LVT %, respectively.

Table 2.19: Results for our heuristic sizing algorithm. $Const3$ is used for the minimum implant width constraint in this experiment.

Benchmarks	WS (ps)			Leak (mW)			Δ #Vio. (%)			CPU total / tool (min)		
	Heur1	Heur2	Heur3	Heur1	Heur2	Heur3	Heur1	Heur2	Heur3	Heur1	Heur2	Heur3
DMA	3	131	**4**	0.046	0.048	**0.047**	64 (33)	50 (26)	**0 (0)**	7 / 5	5 / 3	**7 / 5**
MPEG2	3	19	**19**	0.298	0.315	**0.309**	376 (54)	164 (24)	**9 (1)**	43 / 41	27 / 26	**34 / 33**
AES	6	18	**18**	0.184	0.213	**0.203**	1734 (151)	814 (71)	**412 (36)**	98 / 96	51 / 50	**69 / 68**
JPEG	-5	9	**10**	1.898	2.109	**1.954**	4861 (62)	2921 (37)	**659 (8)**	1209 / 1207	663 / 662	**1093 / 1091**

Experiment 1: Evaluation of MinIA-fix algorithms. Our first experiment evaluates the MinIA-fix algorithm under power and timing constraints. In Table 2.18, *Commercial P&R* indicates the result of simple filler cell insertion performed by a commercial P&R tool. *Δ#Vio.(%)* shows the absolute (relative) change in number of MinIA violations compared to the original number of violations (negative numbers indicate that the number of violations is reduced). *Commercial P&R* does not change the design, and the runtime is almost zero. However, it fixes only 36% of MinIA violations in the worst case (i.e., 64% of violations remain), and 74% on average across all testcases. By contrast, our heuristic substantially reduces the number of MinIA violations (97% in the worst case, and by 99% on average). Note that the WS of all testcases is positive even though Δ*WS* is negative for the case of MPEG with $Const3$, *AES_var2* and *AES_var3*. *Fill (%)* indicates the portion of the total area occupied by filler cells. We see that the numbers are very small, which means that whitespace is not all consumed by filler cells. The *CPU total/tool* shows the total runtime, and the time consumed by the socket interface between external tools (i.e., P&R and timer tool) and our optimizer. Nearly all of the total runtime is consumed by external tools, since the operations used during the optimization such as adding filler cells, moving and/or sizing a cell take *O(few seconds per operation)*.

Experiment 2: MinIA-aware sizing. Our second experiment evaluates three approaches – free sizing (**Heur 1**), restricted sizing (**Heur 2**) and MinIA-aware sizing (**Heur 3**) with our heuristic approach (Algorithm 7) in terms of leakage power, timing and the number of MinIA violations. *Const3* is used for the minimum implant width constraint. Table 2.19 shows **Heur 1**, **Heur 2** and **Heur 3** results after sizing. Although the leakage power values are smallest with **Heur 1**, the increase in number of MinIA violations is up to 151% of the original number of MinIA violations. With **Heur 2**, the leakage power values are high since the sizing is prevented from creating any violation for the target cell. The increase in number of violations comes from the impact of sizing on neighbor cells. **Heur 3** shows near-zero or

small increase in MinIA violations with less leakage power than **Heur 2**. In the **Heur 3** results, we see that solution quality in terms of leakage and timing is nearly maintained, while the number of MinIA violations is greatly reduced.

Additionally, we have applied our placement heuristic to the results of sizing. For AES, we observe that 22% of MinIA violations still remain in the result of **Heur 1**, while 3-4% of those are left for **Heur 2** and **Heur 3**. Note that the MinIA violations may increase the total area and power even though the initial leakage power might be less with **Heur 1**.

2.2.5 Conclusion

In this work, we have addressed a new gate sizing/V_t-swapping and placement problem with the *minimum implant area* (MinIA) constraint. The MinIA constraint presents a new challenge to the physical implementation flow in sub-22nm technology, and requires true co-optimization of placement and gate sizing/V_t-swapping. We have proposed sizing and placement heuristics that optimize power and fixes MinIA violations while minimizing placement perturbation. Compared to commercial P&R tools, our methods achieve significant reductions in the number of MinIA violations under timing/power constraints.

Our current heuristics cannot guarantee to minimize the perturbation of placement and/or the number of violations, though they are straightforward and easy to apply. Hence, similar to [56], we intend to study the use of dynamic programming to solve single-row placement with MinIA fixing. Procedure *FixRowMinImpVio()* in Algorithm 8 sketches such an approach. Our ongoing work includes implementation of, and analysis of results from, dynamic programming based optimizations.

Algorithm 8 Dynamic Programming-based $MinIA$-aware Row Placement

Procedure $FixRowMinImpVio(\{I_{min}\}, R, \{T\})$
Input : minimum implant constraints $\{I_{min}\}$, a placement of a standard-cell row R, a set of timing constraints $\{T\}$
Output : a sizing/placement solution for R
$\{c_i.sol\} \leftarrow \emptyset$;
// $\{c_i.sol\}$ = sizing/placement solutions from the left-most cell to the ith cell
for $i = 1$ to k **do**
  for all $\{l, v, s\}$, where $l = -W$ to W, $v =$ LVT, NVT, HVT, $s = -S$ to S **do**
    // l = Δ cell location, v = V_t, s = Δ cell size
    $Cost(c_{i,l,v,s}) \leftarrow \min_{j \in \{c_{i-1}.sol\}} Cost(\{c_{i,l,v,s}, j\})$
    // $Cost(*)$ = cost of power and minIA violations
    $\{c_i.sol\} \leftarrow \{c_i.sol\} \cup c_{i,l,v,s}.sol$;
  end for
end for
$sol \leftarrow$ the minimum cost solution in $\{c_k.sol\}$;
return sol;

2.3 Heuristic Methods for Fine-Grain Exploitation of FDSOI

Fully Depleted Silicon On Insulator (FDSOI) is a promising process technology especially for low power IoT designs due to its low-cost and low-power potential. Particularly, contrary to the FinFET process in which body bias is inefficient, body bias in FDSOI is a good knob for speed and leakage optimization. At a $0.5V$ supply, speed can be improved by up to $5.5\times$ by using forward body bias (FBB), and leakage power can be reduced by up to $50\times$ by using reversed body bias (RBB) [45].

Due to the unique FDSOI device structure that does not have body, regular V_t (*LR*) and low V_t (*LL*) devices are implemented by a special structure called a *flip well* configuration.[16] Figure 2.10 shows the structures of a conventional well (LR) and a flip well (LL). In the flip well structure, an N-well is implemented under the NMOS transistor, and a P-well is implemented under the PMOS transistor; this structure enables a wide range of forward body bias [75]. However, as the wells are flipped, abutting LL and LR cells induces a well bias conflict. Thus, LL and LR cells must be isolated from each other.[17]

Successful implementations of mixed-V_t and body bias in FDSOI are well-documented in several previous works [14][45][75]. In these works, the V_t and body bias are assigned in a *coarse-grained*, i.e., block-level, manner due to the constraint that different-V_t cells cannot be abutted. To complement these previous works, and to ensure that the V_t and body bias options in FDSOI are fully exploited, *fine-grained* implementations should also be considered and evaluated. However, it is not straightforward to estimate the benefits of fine-grained mixed-V_t and body bias implementation due to the placement constraints for different-V_t options. Indeed, our present work shows that in the FDSOI context, benefits realized from fine-grained use of V_t (hence, body bias) options appears strongly dependent on designs, libraries, performance targets and power requirements. For example, our results and discussion below suggest that realizable benefits depend on (i) availability of rich cell library options that offer power-delay tradeoff, such as poly-biasing options; (ii) the optimized timing structure produced by traditional physical implementations, including aspects such as multiplicity of timing-critical paths ("wall of slack") and the

[16] The names LL and LR map to nomenclature such as LVT/SLVT or HVT/RVT used in popular foundry technologies. In this work, we generically use LL/LR to avoid the use of any foundry-specific names.

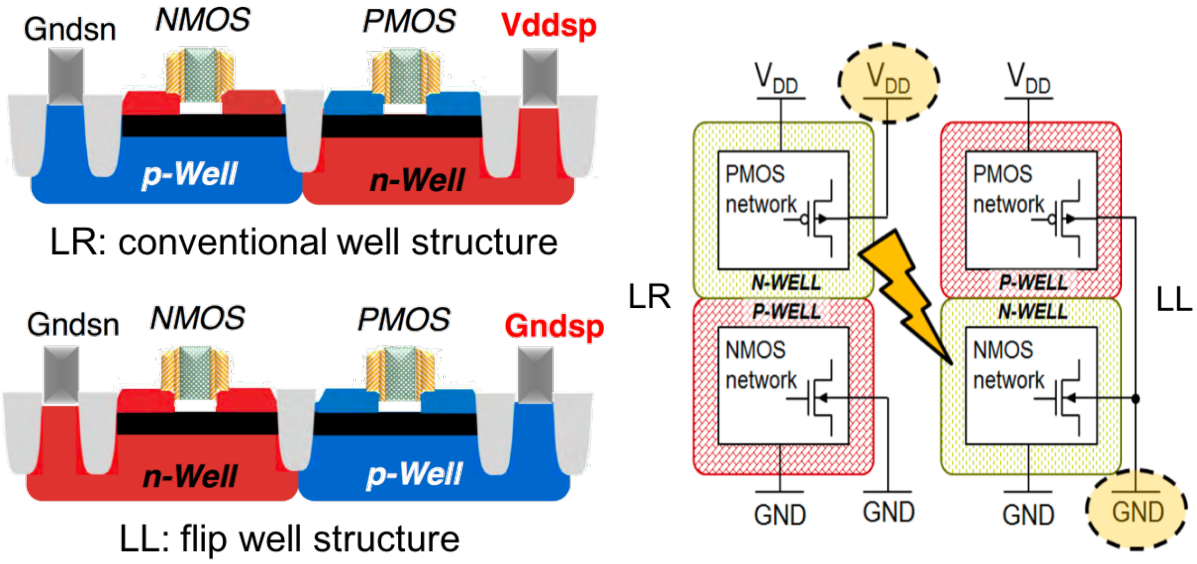
[17] There exist some foundry technologies which offer different V_t with the same well structure (e.g., 22FDX from *Global Foundries* [178]), enabling different-V_t devices to be mixed freely. However, in this work, we refer to "mixing different well structures" as mixed-V_t: we study the regime where different V_ts are generated by using different well structures. Specifically, LL (resp. LR) is formed by a flip (resp. conventional) well. This corresponds to widely-used $28nm$ commercial offerings.

spatial distribution of critical instances; and (iii) a given design's sensitivity to active leakage and dynamic power.

In this work, we study the potential of *fine-grained*, i.e., subblock-level, mixed-V_t assignment in FDSOI. We frame our study using a novel *speed domain partitioning* (SDP) problem formulation, since the V_t assignment essentially seeks to partition the input block into fast and slow parts. Note that in the following, we focus on the challenge of fine-grained mixed-V_t assignment in FDSOI, and we discuss only briefly in Section 2.3.5 the (more difficult) challenge of fine-grained body bias assignment. Both of these challenges share the fundamental problem of determining rectilinear layout regions for speed boost, i.e., performance improvement, with minimum power increase. Moreover, the mixed-V_t assignment problem that we study can be seen as closely related to body bias assignment - e.g., FBB assignment can be comprehended by adding more timing constraints for the FBB mode.

Our main contributions are summarized as follows.

- We formulate the SDP problem and develop two basic optimization flows to address the SDP problem: an ILP-based flow, and a sensitivity function-based heuristic flow.
- We experimentally study the potential benefits of fine-grained mixed-V_t in FDSOI. Up to 20% (resp. 7%) speed improvement with 54% (resp. 26%) LL region area is achieved for an example with generic (resp. "rich") cell library. This said, we observe that outcomes are highly dependent on available library cells as well as characteristics of the input designs.
- We analyze root-cause challenges to fine-grained mixed-V_t exploitation in FDSOI. Specifically, we identify three intrinsic difficulties: (i) availability of rich library cell options; (ii) the existence of a "slack wall" in well-optimized designs; and (iii) spatial contiguity constraints on the placement.
- We suggest a *decision tree* to help assess the potential benefits for a given design of using mixed-V_t in FDSOI.
- Finally, we also explain additional difficulties of fine-grained body bias assignment in FDSOI.



(a) Device structure.

(b) Well conflict.

Figure 2.10: (a) The FDSOI device structure. (b) The LL and LR cells cannot be abutted due to well bias conflicts [45].

2.3.1 Related Work

We now review related works. To the best of our knowledge, there are not many works that directly address fine-grained mixed-V_t and body biasing in FDSOI. Thus, along with fine-grained body bias work, we review previous works in multiple-Vdd (voltage island) placement as well as multiple-height cell placement. The three problems have similarity in that cells are assigned to certain attributes, with consideration of placement constraints, to optimize design speed and power.

Island generation for dual Vdds. A post-placement Vdd assignment flow to minimize the number of level converters is proposed in [51]. Sensitivity-based Vdd assignment is followed by placement optimization based on soft clustering using a min-cut placer to generate voltage islands with a reduced number of level converters. Liu et al. [107] propose a voltage island generation method in placement for dual-Vdd designs. The proposed flow starts with power- and timing-driven placement. Sensitivity-based voltage assignment is performed followed by partition-based placement refinement with soft clustering of the same Vdd cells. During the placement refinement stage, neighboring bins are merged to create a new larger bin, and this new bin is repartitioned considering wirelength and clustering for voltage isolation.

This process is performed iteratively until all the same-Vdd cells are clustered.

Island generation for multiple Vdds. Wu et al. [158] propose a dynamic programming (DP)-based methodology to group voltage islands for designs with multiple supply voltages. The voltage island grouping problem is formulated as follows. Given a set of minimum Vdd assignment v_g for each grid g, and an error threshold δ, find a partitioning with the smallest size (i.e., number of islands) where each island has an error of at most δ. The error of an island I is defined as $\sum_{g \in I} (v_{max} - v_g)$, where $v_{max} = max_{g \in I} v_g$. The heuristic method in [158] has two steps: (i) size reduction to a $p \times q$ array G where each grid has an error less than δ so that the array is manageable by DP-based approach, and (ii) applying DP to G. In the work of [33], a greedy heuristic approach for rectangular voltage island generation is proposed for multiple-Vdd designs. The largest rectangular regions with the minimum resulting power are selected iteratively for a given placement along with grid-based voltage assignment.

Row-based dual Vdds. Yeh et al. propose cell layout techniques along with a simulated annealing-based placement algorithm that support row-based dual-Vdd designs [165]. The authors of [159] propose an improved placement algorithm that handles local clock buffers (LCB) for the row-based dual-Vdd designs. The proposed flow consists of two stages: (i) clustering gates to form voltage islands, and (ii) linear programming (LP)-based legalization. In the work, latches are grouped based on maximum weighted matching. Then, gates are again clustered based on their distance-based weights, followed by min-cost max flow-based level shifter assignment. The authors of [160] propose an extension of [159] that improves timing by moving gates for the row-based dual-Vdd designs. The proposed flow moves timing-critical gates to feasible locations in a greedy manner, without changing voltage assignments.

Mixed-height cell placements. The work of [37] proposes a placement optimization flow to implement a fine-grained non-integer multiple-height cell placement. DP-based partitioning is followed by sensitivity-based gate sizing and placement optimization in the proposed flow.

Other fine-grained body bias work. Flores [46] proposes a greedy algorithm that determines a body bias island floorplan that gives minimum wirelength. In the proposed algorithm, the size and location of the islands are selected based on track utilization. Taco et al. [147] study gate-level dynamic body biasing in $28nm$ FDSOI in the circuit level. The authors of [147] compare the dynamic threshold voltage MOSFET circuit (transistor-level body biasing) and the gate-level body-biased circuit [4][32]. Kühn et al.

[95] propose a body bias domain partitioning method by identifying gates activated through a common identifier during logic synthesis. The authors of [95] assign body bias to partitioned domains based on leakage and timing.

In sum, previous works have addressed design optimizations in contexts – notably, voltage island and mixed-height placement – that are similar to our present FDSOI context. Many of these other works report noteworthy power/speed/area benefits from their proposed optimizations. However, as seen from our experimental results as well as the Section 2.3.4 discussion below, the SDP problem in FDSOI seems fundamentally more challenging for several reasons.

2.3.2 Our Approach

In this section, we first formulate the SDP problem for mixed-V_t FDSOI implementation. We then describe a set of implementation approaches we have tried, and give details of the two best approaches that empirically give maximum speed improvements (for given power overhead) subject to placement constraints.

It must be emphasized that there are many conceivable ways to implement fine-grained mixed-V_t (and body bias) implementations, working at various design levels in the RTL-to-GDS flow. For example, one could plausibly synthesize with both V_ts and partition the netlist according to the V_t of each gate. Or, one could synthesize with LR-only and optimize the netlist while making LL cells additionally available. Among many possible implementation flows that we have investigated, our discussion focuses on *post-placement* optimizations in which we start from placed netlists *implemented with LR-only cells*, without awareness of placement constraints in FDSOI. We have zeroed in on this space of implementation flows, for the following reasons. First, it is difficult to identify timing-critical cells that will eventually require speed boost, based on a pre-placement netlist – since the timing changes disruptively after placement. Second, if we predetermine (LL) regions for speed boost, this restricts placement and leads to poor quality of placement results with respect to timing and wirelength. Third, if we allow the mixing of LL/LR V_t values up front in synthesis, the optimization of LL vs. LR regions becomes highly restricted by existing (placements of) LL cells; in our experience, this leads to very large timing and/or power penalties.

Mixed-V_t Speed Domain Partitioning ("SDP-MVT") Problem

Given an initial placed design implemented with LR cells only, perform V_t swapping, sizing and placement optimization to define LL regions under timing/placement constraints.

Input: A placed design, synthesized and optimized with LR cells.

Output: An optimized mixed-V_t netlist/placement, with LL islands.

Constraints: The optimization would typically be subject to timing and placement constraints. For example, the target operating frequency of the mixed-V_t implementation should be X% higher than that of the pure-LR implementation; the total power of the mixed-V_t implementation should be no more than Y% higher than that of the pure-LR implementation; and the total area of LL regions should be no more than Z% of the total area of the design. Our experiments below attempt to shed light on achievable combinations of X, Y and Z for the testcases studied.

Overall Flow

Figure 2.11 illustrates our overall flow. The input to the overall flow is a placement implemented and optimized with LR cells only. We generate LL islands and assign LL to all the cells in the islands. We then further optimize timing if needed without changing the LL assignment. The output is an optimized placement (and netlist) with LR and LL cells. Our background experiments have tried a rather large number (i.e., $4 \times 5 = 20$) of flow variants with various combinations of island generation methods and timing optimization methods. Additionally, various alternative ILP-based assignment strategies have been tested.

For the **island generation**, we have tried four flows, namely, (i) a "brute-force" approach; (ii) sensitivity-based assignment; (iii) ILP-based assignment; and (iv) iterative heuristic-based assignment. In the brute-force approach, we first collect all the timing-critical cells in a given placement, and move these cells into a predefined rectangular region. We sweep the locations and aspect ratios of the predefined region to find a best solution; thus, the brute-force method is similar to executing the method of [46] with many different target regions. The sensitivity-based assignment is performed as follows, with respect to a coarse gridding of the layout region into ($\sim$100) grids. First, for each cell in the layout, we calculate the sensitivity (1/Δ total negative slack) when the LR cell is changed to LL. We then select the top-k% of all

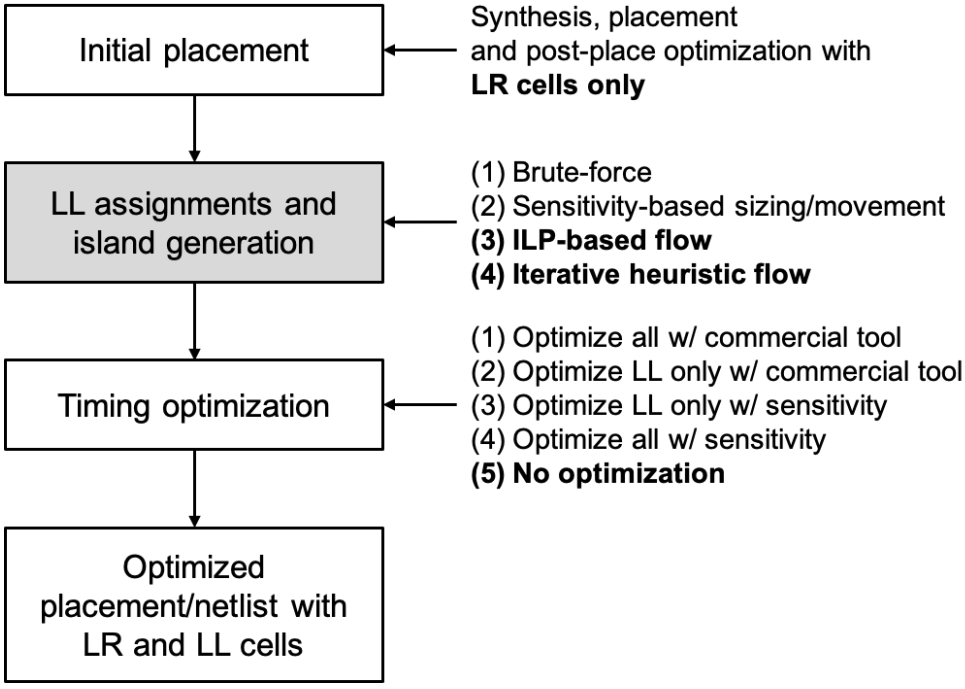


Figure 2.11: Overall flow, showing options explored in background studies. Superior flow options are shown in bold font.

cells with respect to the sensitivity. Among the selected cells, we randomly choose n cells to use as seeds for the LL region generation. (Since LL regions must be contiguous in the final layout solution, succeeding LL cells should be close to these seed LL cells.) Using the chosen n cells as "anchors", we recalculate the sensitivity of each cell considering proximity, i.e., the quotient (distance to the nearest anchor / Δ total negative slack). We then sum all of the cell sensitivity values in each grid. Finally, we assign LL to the regions (grids) with the largest sums of sensitivity values.

For the **timing optimization**, we have tried five flows, namely, (i) commercial tool-based optimization for islands where we run post-placement optimization for the LL region only; (ii) commercial tool-based optimization for the entire placement; (iii) sensitivity-based gate sizing and movement for islands; (iv) sensitivity-based gate sizing and movement for the entire placement; and (v) no optimization. For the sensitivity-based gate sizing (iii), we estimate Δ slack for each potential move and swap. Δ slack / current slack is used as our sensitivity function.

From our experimental studies, we have concluded that ILP-based assignment and iterative

heuristic-based assignment offer clearly superior results in terms of speed improvement for given placement constraints and power overhead. For timing optimization, none of the methods studied is helpful to improve worst setup slack. In sum, the two best overall flows that we select for our experiments below are designated in bold font in Figure 2.11.

Algorithms for Island Generation

We now give details of the two approaches for island generation that offer the best results in our experiments. In our island generation flows, we perform grid-based assignment, i.e., all the cells in the same grid have the same V_t. Thus, island generation first splits the layout into a number of uniform rectangular grids. Empirically, the results of our flows are not sensitive to the number of grids when this number is 100 or more. Thus, in the following we report results for 100 uniform rectangular grids.

Table 2.20: Notations.

Notation	Meaning
α	LL island area constraint (0 – 1)
N	maximum number of islands
i / j / m / n	index of cell / LL island / timing path / grid, respectively
v_n^j	binary indicator of whether grid n belongs to island j
v_j	binary indicator of whether island j is generated
g_i	grid index of cell i
Δd_i	delay difference between LR and LL for cell i LR cell delay subtracted by LL cell delay.
s_m	negative slack of path m
d_m	initial delay of path m
cp	clock period
max_area	maximum area constraint for islands
x_l^j, y_l^j (resp. x_u^j, y_u^j)	x, y locations of lower-left (resp. upper-right) corner of island j
x_l^n, y_l^n (resp. x_u^n, y_u^n)	x, y locations of lower-left (resp. upper-right) corner of grid n
x_l, y_l (resp. x_u, y_u)	x, y locations of lower-left (resp. upper-right) corner of the layout
G	large number

ILP-Based Flow. In our ILP-based flow, we first collect timing critical paths and formulate timing constraints. Table 2.20 shows the notations used in our ILP formulation. The objective is to minimize the clock period (Equation (2.2)). Equation (2.3) ensures that total area of islands is not more than a given maximum area. The area of a rectangle is estimated by its half perimeter. Equation (2.4) ensures at most

one island to be selected for a grid. Equation (2.5) ensures that the number of islands does not exceed N. Equation (2.6) ensures the delay improvement of swapping to LL to be larger than $-s_m$ for each timing path m. Equation (2.7) determines the dimension of islands such that every LL grid is covered by LL islands.

$$\textbf{Minimize: } cp \tag{2.2}$$

Subject to:

$$\sum_j (x_u^j - x_l^j + y_u^j - y_l^j) < \alpha \cdot (x_u - x_l + y_u - y_l) \tag{2.3}$$

$$\sum_j v_n^j \leq 1, \ \forall n \tag{2.4}$$

$$v_j \geq v_n^j, \ \forall n \qquad\qquad \sum_j v_j \leq N \tag{2.5}$$

$$p_m - \sum_{i \in path_m} (\sum_j v_{g_i}^j) \cdot \Delta d_i < cp, \ \forall m \tag{2.6}$$

$$x_u^j - G v_n^j + G > x_u^n, \ \forall j$$

$$x_l^j + G v_n^j - G < x_l^n, \ \forall j$$

$$y_u^j - G v_n^j + G > y_u^n, \ \forall j$$

$$y_l^j + G v_n^j - G < y_l^n, \ \forall j \tag{2.7}$$

Sensitivity Function-Based Heuristic Flow

Algorithm 9 shows our sensitivity function-based heuristic flow for LL island generation. We select LL island regions based on sensitivity, where the average slack of the region as the sensitivity function (SF). The island regions are comprised of contiguous grids. In Line 1, we find a small rectangular region based on sensitivity, and use this as a seed for generating island regions. In Lines 2 – 12, we grow the seed gradually until the area of the region reaches a given max_area constraint. More specifically, for each direction (Line 3), we try growing the seed region by one grid (Line 4) and calculate SF (Line 5). We select the best direction to grow, which gives the best (minimum) SF score (Line 7).

Algorithm 9 Sensitivity function-based heuristic flow.

Procedure: $HeurV_tAssign()$
Input: placement, max_area
Output: placement with LL islands

```
region ← FindBestRegion(area=0.01, SF)
while area < max_area do
    for direction in (east, west, north, south) do
        grow_region ← grow(region, direction)
        score ← CalcSF(grow_region, SF)
        if best_score < score then
            region ← grow_region
        end if
    end for
    area ← region.area
    update timing
end while
return region
```

2.3.3 Experimental Setup and Results

In this section, we report the results of our two best flows that we describe in Section 2.3.2. For each of our testcases, we perform three distinct optimizations to find rectilinear regions for the mixed-V_t problem: (i) ILP with one island (**ILP-1**); (ii) Iterative ILP with two islands (**ILP-2**); and (iii) Heuristic with two islands (**Heur**). The Iterative ILP simply defines a first LL island based on the above-described ILP, and then – based on this LL assignment – sets up and solves the same ILP with updated timing information to define a second LL island. Our methods are implemented in Tcl 8.4 [199], and CPLEX v12.6.3 [180] is used as the ILP solver. Runtimes for each solution, including runtimes of commercial tool steps and any CPLEX runtimes, are at most 6 hours on a 2.8GHz Xeon server with 128GB RAM.

Experimental Setup

We perform experiments in a $28nm$ FDSOI foundry technology with dual-VT libraries, $0.9V$ nominal supply voltage. We validate our flows with ARM Cortex M0 and M3 cores, along with two designs (ldpc, viterbi) from the OpenCores website [191].

The designs are synthesized with target periods in the range of $0.5ns$ – $2.5ns$, with a $50ps$ step. For each synthesized netlist, P&R is performed with three P&R target periods, i.e., synthesis target

period + $\{$-50, 0, 50$\}ps$. The SP&R (synthesis and P&R) is performed with three library options, i.e., LR-only, LL-only and LR+LL (mixed V_t), without placement constraints. For the P&R flow, high leakage power optimization effort is applied, along with post-placement leakage optimization. For each SP&R implementation, we record target P&R clock period + worst negative slack (WNS), namely, *effective clock period*, along with leakage power and total power. (Assuming a simple single clock constraint, worst setup slack plus the target clock period is equal to the smallest clock period at which setup slack = 0. We refer to this as the effective clock period (ECP).)

We have performed our experiments with multiple commercial $28nm$ FDSOI enablements, one with 12-track (12T) cells and a "generic" (no poly biasing) library, and the other with 8-track cells and a "rich" library with poly biases.[18]

- **Enable1.** $28nm$ 12T LL and LR without poly bias (i.e., P0) are used as library cells. Synopsys Design Compiler N-2017.09 [195] and Cadence Innovus v17.1 [174] are used for logic synthesis and P&R tools, respectively.

- **Enable2.** $28nm$ 8T LL and LR with four poly bias options (i.e., P0, P4, P10 and P16) are used as library cells. Cadence Genus v16.2 [173] and Cadence Innovus v15.2 [174] are used for logic synthesis and P&R tools, respectively.

We have implemented the M0 and M3 testcases with each of these two enablements. We denote M0 (resp. M3) implemented with Enable2 as M0-2 (resp. M3-2).

Table 2.21 shows the testcase information. All the numbers are reported by the P&R tools that correspond to the associated enablements (i.e., Innovus v17.1 and v15.2 for Enable1 and Enable2, respectively), at the post-placement stage. The *Min. LL period* and *Min. LR period* columns show the minimum achievable effective period with LL-only and LR-only SP&R implementation, respectively. We observe that roughly 30$\sim$40% speedup can be achieved with LL-only implementations (i.e., 100% LL), compared to LR-only implementations. This can be viewed as an upper bound on speedup that could be achieved by any mixed-V_t implementation.

[18]Poly biasing refers to transistor gate (channel) length biasing, typically by a positive number of nanometers, for ultra fine-grain exploitation of leakage-delay tradeoff. For example, P10 denotes a +10nm (relative to the nominal value) channel length.

Table 2.21: Testcases.

Enable	Name	#Instances	Min. LL period	Min. LR period
Enable1	M0	7K∼11K	0.803	1.122
	ldpc	46K∼59K	0.775	1.027
	M3	47K∼60K	1.151	1.599
	viterbi	53K∼69K	0.529	0.766
Enable2	M0-2	8∼13K	0.834	1.144
	M3-2	47∼71K	1.248	1.669

Experimental Results

Recall that we perform three optimizations, i.e., ILP-1, ILP-2 and Heur, on LR-only implementations to find rectilinear regions for mixed-V_t FDSOI implementation. To obtain meaningful inputs to our optimization flows, we select several LR-only implementations with effective clock periods no more than 1.1 $\times$ *Min. LR period*. We run our optimizations with maximum LL% constraints of $\{20\%, 30\%, 40\%, 50\%\}$. We then report the results with minimum leakage power while meeting $\{2\%, 5\%, 7\%, 10\%, 20\%\}$ speed improvements compared to *Min. LR period*.

Table 2.22 shows the results of our three heuristics for mixed-V_t optimization (ILP-1, ILP-2, Heur), along with three reference implementations: the LR-only (*LR*) baseline implementation, the LL-only (*LL*) fastest-possible implementation, and the mixed-V_t (without placement constraints) (*LR+LL*) implementation that bounds the mixed-V_t power-speed tradeoff. In each group of columns of the table, we report parameters (effective clock period, leakage and total power, and percentage area of LL region(s)) of the lowest-power solution that achieves the given percentage speed improvement, relative to the LR baseline implementation. (Note that for each testcase, the 'LR' numbers are the same in each group of table columns.) Blanks (indicated by '-') mean that there is no result that meets the corresponding speed improvement with the corresponding flow. We note that our target periods are based on *Min. LR period* which are aggressive, and that the upper bound of the possible speedup is only 30∼40%. We also note that we do not include the implementation overhead due to the spacing rules between LL and LR. In reality, mixed-V_t implementations with placement constraints will have more area and thus consume larger total power. Figure 2.13 shows the leakage and total power versus the effective clock periods for the four designs implemented with Enable1, and for the two designs implemented with Enable2. Figure 2.12 shows

the island shape obtained by ILP-2 for the viterbi testcase in Enable1. This obtains a speedup of 20% with LL region area of 54%.

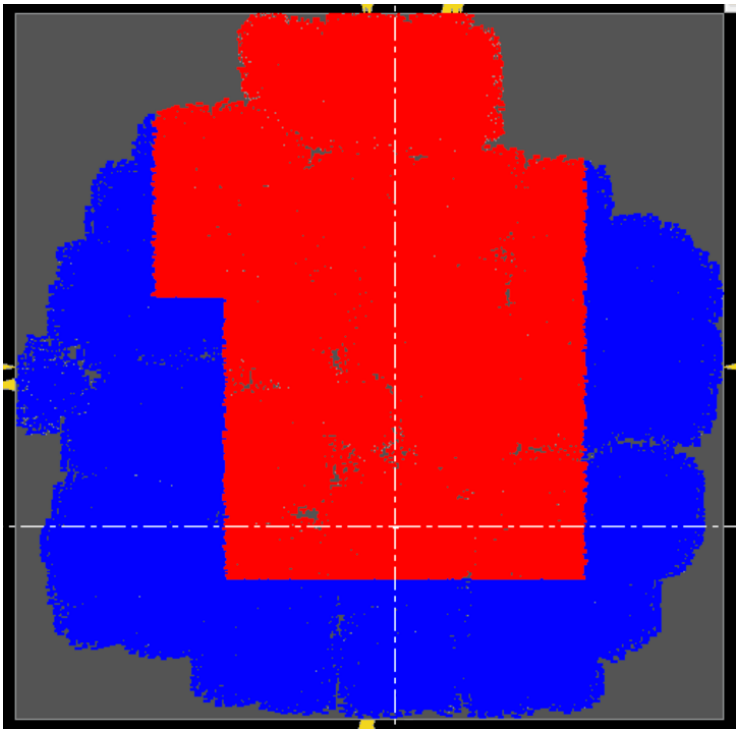


Figure 2.12: Island shape obtained by ILP-2 for the viterbi testcase in Enable1. LL regions are highlighted in red.

Our high-level findings are summarized as follows.

- The overall outcomes, i.e., power and speed benefits from mixed-V_t, are strongly library- and design-dependent.
- For Enable1 (generic library)-based designs, up to 20% speed improvement with 54% LL region is observed.
- For Enable2 (rich library)-based designs, up to 7% speed improvement with 26% LL region is observed.

Designs with generic library cells (Enable1). For the designs implemented with Enable1, we observe that *M0* and *viterbi* achieve 20% speedup with ILP-based optimizations. For *M0*, ILP-1 achieves the *20% speedup* with 53% LL area. Compared to *LR+LL* (without placement constraints) in the ILP-1

solution, leakage and total power values are 48% and 18% larger while the LL area is the same. For *viterbi*, ILP-1 and ILP-2 achieve the 20% speedup with 60% and 54% of LL area, respectively. Compared to *LR+LL*, in the ILP-2 solution, leakage power value is 69% larger, but total power value is 1% smaller, while LL area is 25% larger. For *ldpc*, ILP-2 achieves the 7% speedup target with 53% of LL area. Compared to *LR+LL*, in the ILP-2 solution, leakage and total power values are 101% and 20% larger, respectively, while LL area is 3% smaller.

We see that *ldpc* is not a SDP-friendly design, since the LL portion in *LL+LR* is higher compared to other designs. Even without placement constraints, $\sim 77\%$ of area is needed to achieve the 20% speedup requirement. For *M3*, ILP-2 achieves the 7% speedup with 53% LL area. Compared to *LR+LL*, in the ILP-2 solution, leakage and total power values are 216% and 119% larger, while LL area is 35% larger. We believe that there are at least two reasons for not achieving $\geq 10\%$ for *M3*. First, since *M3* has a relatively larger initial effective clock period compared to other testcases, it is more challenging to achieve a higher % of speedup. In particular, WNS must be improved by more than $250ps$ for the 20% speedup target, while other designs can meet such a speedup goal with less than $200ps$ WNS gain. Second, the placement seems to not be SDP-friendly. The required LL area values to achieve 7% speedup target without and with placement constraints are 18% and 53% respectively, and there is a very large gap (i.e., 35% difference in LL area) between these two cases. This indicates that critical cells are indeed placed sparsely (that is, without any spatial contiguity awareness in physical synthesis or sizing steps) by commercial implementation flows, and that it is not easy to cover the critical cells with only a couple of rectangular LL regions.

In Figures 2.13(a)-(h), we observe that Heur, ILP-1 and ILP-2 curves are between the LL and the LR+LL curves, but closer to the LR+LL curves. However, for faster effective clock periods (i.e., less than the minimum clock period achievable in pure-LR designs), the power values of the Heur, ILP-1 and ILP-2 increase dramatically. We note that the LR+LL results are obtained *without* considering placement constraints while the Heur, ILP-1 and ILP2 consider the placement constraints. The unnecessary cell swap to LL due to the placement constraints leads to large power increase.

Designs with rich library cells (Enable2). For the designs implemented with Enable2, we observe that up to 7% and 2% speed improvement is achieved for *M0-2* and *M3-2*, respectively. In

Table 2.22: Results of ILP-1, ILP-2 and Heur.

Design	Flow	2% imprv				5% imprv				7% imprv				10% imprv				20% imprv			
		ECP	PLeak	PTot	LL%	ECP	PLeak	PTot	LL%	ECP	PLeak	PTot	LL%	ECP	PLeak	PTot	LL%	ECP	PLeak	PTot	LL%
M0	LR	1.102	0.17	8.09	0	1.102	0.17	8.09	0	1.102	0.17	8.09	0	1.102	0.17	8.09	0	1.102	0.17	8.09	0
	LL	1.059	1.15	7.42	100	1.008	1.24	7.96	100	1.008	1.24	7.96	100	0.996	1.31	8.57	100	0.917	1.49	9.59	100
	LR+LL	1.070	0.55	7.05	31	1.025	0.69	7.76	39	1.025	0.69	7.76	39	0.947	0.78	8.23	41	0.911	1.02	9.25	53
	ILP-1	1.053	0.39	8.14	15	1.042	0.91	9.11	30	1.007	1.05	9.28	41	0.992	1.27	9.66	52	0.915	1.51	10.95	53
	ILP-2	1.046	0.59	8.51	19	1.046	0.59	8.51	19	0.990	1.07	9.53	40	0.990	1.07	9.53	40	-	-	-	-
	Heur	1.045	0.45	8.25	17	1.045	0.45	8.25	17	1.027	1.13	9.47	44	1.000	1.48	10.02	54	-	-	-	-
ldpc	LR	1.027	1.17	79.44	0	1.027	1.17	79.44	0	1.027	1.17	79.44	0	1.027	1.17	79.44	0	1.027	1.17	79.44	0
	LL	0.972	7.58	89.37	100	0.972	7.58	89.37	100	0.941	8.49	91.21	100	0.916	8.75	86.74	100	0.820	12.41	108.42	100
	LR+LL	0.962	4.55	78.07	50	0.962	4.55	78.07	50	0.909	5.63	86.05	56	0.909	5.63	86.05	56	0.817	10.06	104.42	77
	ILP-1	1.003	5.51	85.49	22	0.977	11.31	97.12	53	-	-	-	-	-	-	-	-	-	-	-	-
	ILP-2	0.996	4.14	84.53	16	0.967	9.13	93.20	43	0.954	11.32	102.89	53	-	-	-	-	-	-	-	-
	Heur	1.004	7.24	88.39	40	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
M3	LR	1.577	0.75	29.87	0	1.577	0.75	29.87	0	1.577	0.75	29.87	0	1.577	0.75	29.87	0	1.577	0.75	29.87	0
	LL	1.479	6.53	36.10	100	1.479	6.53	36.10	100	1.403	7.08	38.88	100	1.403	7.08	38.88	100	1.295	8.03	43.83	100
	LR+LL	1.506	2.05	31.88	18	1.457	2.19	33.81	18	1.457	2.19	33.81	18	1.433	2.37	33.72	20	1.288	5.44	44.43	43
	ILP-1	1.524	1.64	32.99	8	1.492	6.50	38.91	47	-	-	-	-	-	-	-	-	-	-	-	-
	ILP-2	1.516	2.79	34.46	18	1.493	3.52	35.67	21	1.459	6.92	40.21	53	-	-	-	-	-	-	-	-
	Heur	1.536	2.48	33.77	17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
viterbi	LR	0.755	1.63	157.37	0	0.755	1.63	157.37	0	0.755	1.63	157.37	0	0.755	1.63	157.37	0	0.755	1.63	157.37	0
	LL	0.733	15.55	165.62	100	0.700	15.99	175.61	100	0.700	15.99	175.61	100	0.664	16.32	188.49	100	0.615	16.86	201.46	100
	LR+LL	0.740	4.71	157.86	10	0.710	7.03	170.08	21	0.643	7.03	187.43	20	0.643	7.03	187.43	20	0.603	8.97	203.19	29
	ILP-1	0.719	6.89	167.12	25	0.719	6.89	167.12	25	0.687	7.51	170.20	36	0.666	10.20	177.87	53	0.611	14.58	202.07	60
	ILP-2	0.715	6.00	167.54	18	0.715	6.00	167.54	18	0.703	6.15	168.45	21	0.664	6.83	179.53	23	0.629	15.23	200.76	54
	Heur	0.698	5.53	172.79	15	0.698	5.53	172.79	15	0.698	5.53	172.79	15	-	-	-	-	-	-	-	-
M0-2	LR	1.144	0.08	5.69	0	1.144	0.08	5.69	0	1.144	0.08	5.69	0	1.144	0.08	5.69	0	1.144	0.08	5.69	0
	LL	1.067	0.37	5.55	100	1.067	0.37	5.55	100	1.067	0.37	5.55	100	1.014	0.52	6.46	100	0.938	0.64	7.61	100
	LR+LL	1.057	0.33	5.69	66	1.057	0.33	5.69	66	1.057	0.33	5.69	66	0.987	0.52	6.72	72	0.897	0.57	7.75	77
	Heur	1.073	0.26	6.42	22	1.073	0.26	6.42	22	1.054	0.33	6.62	26	-	-	-	-	-	-	-	-
M3-2	LR	1.669	0.48	23.92	0	1.669	0.48	23.92	0	1.669	0.48	23.92	0	1.669	0.48	23.92	0	1.669	0.48	23.92	0
	LL	1.555	1.05	21.84	100	1.555	1.05	21.84	100	1.555	1.05	21.84	100	1.477	1.31	23.39	100	1.384	1.63	25.81	100
	LR+LL	1.548	0.71	21.74	43	1.548	0.71	21.74	43	1.548	0.71	21.74	43	1.480	0.93	23.65	49	1.389	1.41	26.42	56
	Heur	1.625	1.44	25.70	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figures 2.13(i)-(l), we see that for designs with Enable2, as noted in Section 2.3.4, the benefit of mixed V_t is relatively less. As expected, the Heur results do not show much power benefit, iso-effective clock period.

2.3.4 On the Difficulty of the SDP Problem

From our experimental results above, we see that the benefit of fine-grained mixed-V_t can be disappointingly small for FDSOI implementations. In this section, we present what we believe to be root-cause, *intrinsic* reasons behind the difficulty of obtaining larger benefits from fine-grained mixed-V_t in FDSOI. These reasons stem from the nature of popular $28nm$ FDSOI foundry technologies, as well as the input designs, that we study. All of these reasons also apply to fine-grained body biasing in FDSOI, as discussed in Section 2.3.5 below.

In this section, our discussion is supported by experimental results obtained through limited access to another commercial enablement, which we refer to as *Enable3*. For Enable3, $22nm$ 8T LL and LR that have six cell variants respectively are used as library cells. Synopsys Design Compiler N-2017.09 [195] and Cadence Innovus v17.1 [174] are used for logic synthesis and P&R tools, respectively.

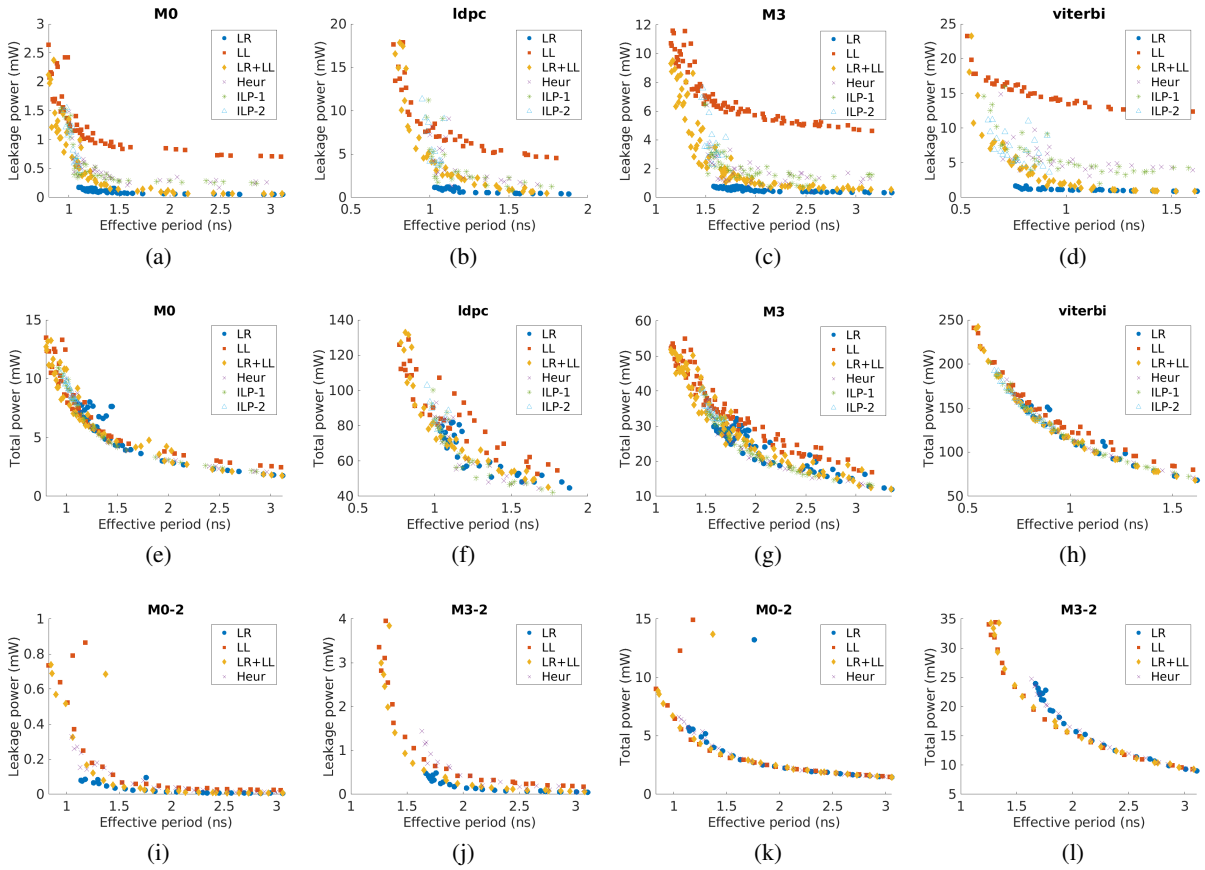


Figure 2.13: (a)-(d) Leakage power versus effective periods for the four designs implemented with Enable1; (e)-(h) total power versus effective periods for the four designs implemented with Enable1; (i)-(j) leakage and (k)-(l) total power versus effective periods for the two designs implemented with Enable2.

Rich Library Cell Options

We observe that the nature of foundry libraries can affect available mixed-V_t benefit. The Enable2 $28nm$ FDSOI foundry enablement offers rich library cell options, i.e., four poly bias (PB) options (P0, P4, P10 and P16) for each group of LL and LR. (The libraries in Enable3 offer six different cell options for each of LL and LR. These options in the same-well-structure group can be mixed in the layout without placement constraints, and they offer power-delay tradeoff wide enough to replace different well structure groups.)

Study of individual cell delays. Figure 2.14 shows leakage-vs.-delay curves for different sizes of buffer cells for each V_t/PB option. The delay value of each cell is calculated using the lookup table in the non-linear delay model (NLDM) library with input slew $50ps$ and load of $4\times$ the input capacitance of each cell. The average leakage power and the delay of each cell respectively correspond to the y-axis and the x-axis in the figure. We observe that the power-delay curves of LL and LR expand with the availability of

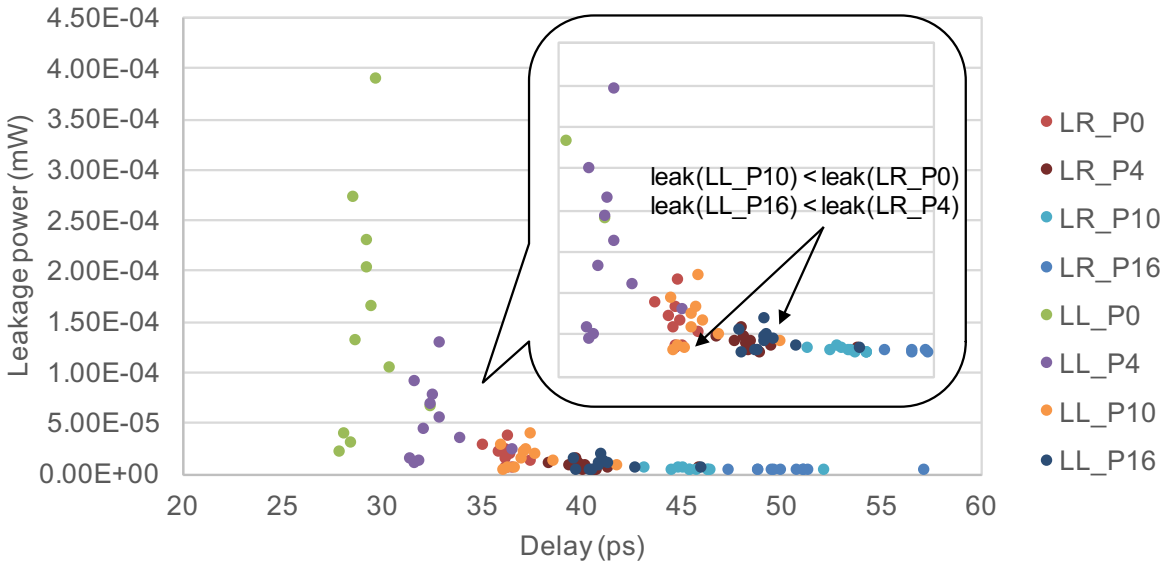


Figure 2.14: Leakage versus delay curves of buffer cells with various V_t and poly bias options available in Enable2. The delay is measured with input slew $50ps$ and output load of $4\times$ the input capacitance of each cell. LL_P16 and LL_P10 consume less leakage power than LR_P4 and LR_P0.

more PB options, such that these curves have greater overlap and become more "near-continuous", with near-minimum power being achievable for a particular operating frequency *without* mixing LL and LR.

Accordingly, the benefit of mixed-V_t is less likely to justify the overheads from placement constraints. (This might be in contrast to multi-Vdd or mixed-height placement contexts, where delay-power tradeoff curves remain disjoint and mixing of flavors retains benefits.) Further, a reversed leakage trend is seen, i.e., leakage power of LL_P16 and LL_P10 is lower than that of LR_P4 and LR_P0 with iso-delay, in the zoomed-in region. With the availability of rich PB options, the benefit of fine-grained mixed-V_t might not be sufficient, let alone compelling, compared to "mixed PB" implementations.

Study of design implementations. Our experimental studies confirm that with rich cell library options, mixing of LL and LR might achieve only limited benefits. For example, Figure 2.15 shows the power and delay tradeoff of different M3 implementations with different library cells, i.e., LL, LR, LR+LL (mixed-V_t) with Enable1, Enable2 and Enable3. Each dot corresponds to a distinct SP&R implementation with a target period. The x-axis shows effective clock periods (ECPs) in ns, which we calculate as the target period subtracted by worst setup slack. Leakage or total power values are shown in the y-axis. We note that Enable1 is an enablement with "generic" cell library options, since only one cell option is available for each of LL and LR.

Notice that when a cell library option is limited, as in the case of Enable1, mixing V_t is beneficial especially for leakage power. However, with rich cell library options (Enable2 and Enable3), the benefit of mixing V_t is less. In Figure 2.15, the red, yellow, blue curves (dots) correspond to LL, mixed-V_t, LR designs, respectively. For relatively slower effective clock periods (i.e., achievable by LR designs), LR always dominates in terms of leakage and total power. For relatively faster effective clock periods (i.e., less than the minimum achievable clock period), mixed-V_t dominates in terms of leakage and total power in Enable1. For Enable1 plots (Figures 2.15(a) and (d)), the gap between the yellow and the red curves dots is clearly visible. However, such a trend is not observed with either Enable2 or Enable3. We also note that the plots in Figure 2.15 do not consider placement constraints. The benefits of LL_LR designs may be obviated if placement constraints are considered. That is to say, when spatial contiguity constraints are considered in the placement, additional V_t swaps must be made to achieve legal placements, and hence the LL region will be larger than necessary.

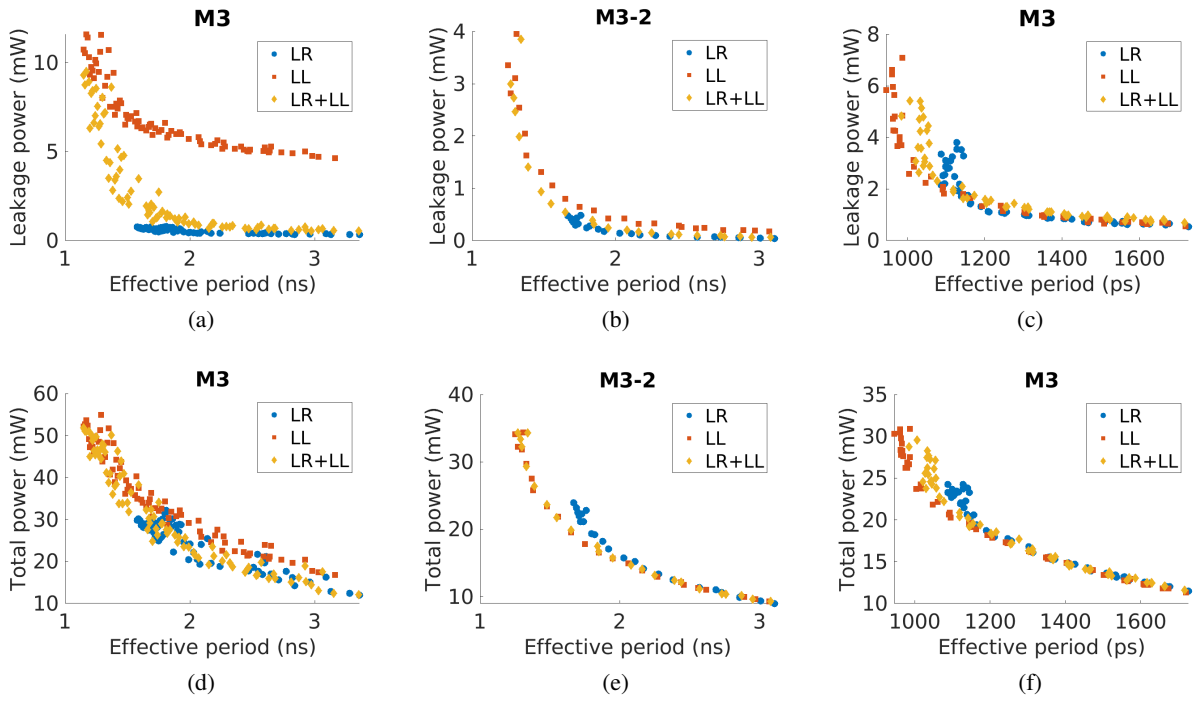
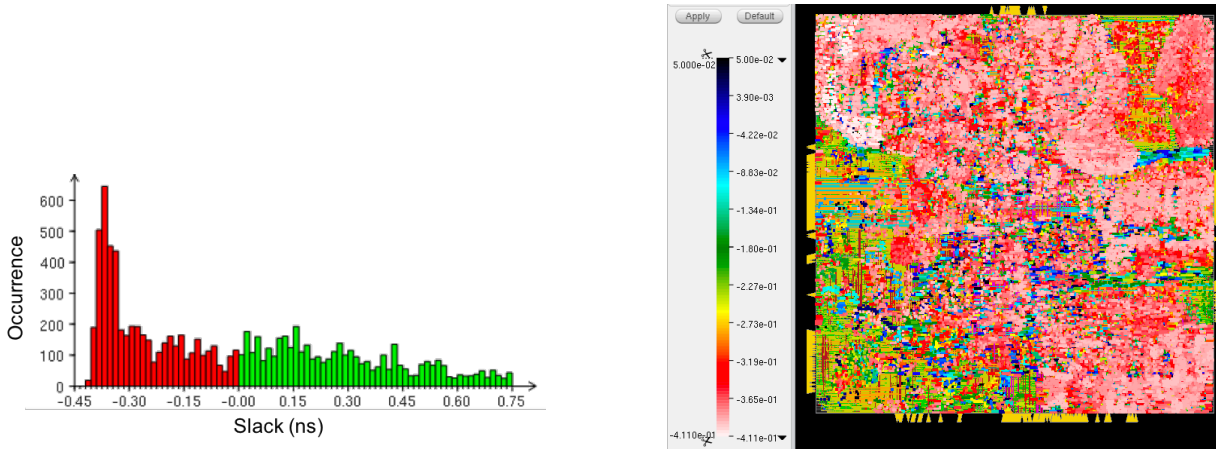


Figure 2.15: Leakage power versus effective period curves for various M3 implementations with (a) Enable1, (b) Enable2 and (c) Enable3, along with total power versus effective period curves for various M3 implementations with (d) Enable1, (e) Enable2 and (f) Enable3.

Many Near-Critical Paths

We also find that speed domain partitioning is difficult to apply to designs with many timing-critical paths. In such designs, a standard physical implementation flow will place timing-critical cells all over the layout. Figure 2.16(a) shows timing statistics of M3 implemented with Enable2. The target clock period of the M3 design is 2.1ns, and timing is measured after post-placement optimization based on trial route. The worst setup slack is approximately -410ps, which makes the effective clock period (i.e., the target clock period subtracted by the worst setup slack) 2.5ns. In the figure, the x-axis and y-axis show the setup slack values and the occurrence of timing endpoints with the corresponding setup slack. We observe a typical "slack wall", namely, that there is a high occurrence of timing endpoints near the worst (leftmost) slack value. Furthermore, due to the nature of optimizers that try to convert timing slack to minimize power, it is likely to see such a slack distribution (slack wall) after post-placement optimization. Designs with higher slack walls are more difficult to improve the worst slack, since more timing endpoints need to be improved

for the clock period to change. E.g., in Figure 2.16, $\sim$30% of timing endpoints should be improved to obtain $\sim 100ps$ slack improvement (which is only 4% of the effective clock period).



(a) (b)

Figure 2.16: Timing information of M3 implemented with Enable2. (a) Histogram of path slack values, showing existence of wall of slack. 30% of paths must be fixed to achieve a 4% speed improvement. (b) Map of instance timing slacks of M3 implemented with Enable2, with legend shown in the left bar. White and red cells are timing-critical.

Placement Constraints

The flip well structure in FDSOI is also a root cause of limited benefit in mixed-V_t implementation. To form rectilinear islands, it is inevitable to make unnecessary V_t swaps: if an LR cell instance must be swapped to LL, the neighbor cells of this target cell must be swapped as well. Figure 2.16(b) shows a timing slack map of M3 with *Enable2*. White and red cells can be considered as timing-critical, as seen in the left legend bar. We observe that the benefit of using LL, i.e., speed improvement, dramatically drops as we give more placement constraints. Based on our experiments, for the M3 design, 9% speed improvement is achievable by swapping 17% of the area to LL without considering placement constraints. However, with placement constraints, the speed improvement drops to 1% with a similar area of LL swaps (19%). We also have studied a variety of pre-placement optimizations, but without success. More specifically, we collect all the timing-critical cells up front and place them locally, i.e., with region constraints. With the recent release of commercial P&R tools that we use, we find that this approach is too disruptive to

conventional timing- and wirelength-driven placement, and that it leads to several suboptimal placement solutions with worse QoR in terms of both timing and wirelength.

2.3.5 Conclusion

In this work, we have studied the potential of fine-grained mixed-V_t optimization in FDSOI. We formulate the speed domain partitioning (SDP) problem and propose effective heuristics that are capable of achieving significant speed improvements. We also identify inherent challenges that limit benefit from fine-grained mixed-V_t: (i) availability of rich cell library options in some commercial foundry enablements; (ii) existence of a slack wall in well-optimized designs; and (iii) spatial contiguity constraints (arising from well structure) in the placement. These challenges are confirmed in implementation experiments with multiple commercial enablements at $28nm$ and $22nm$. Given our observations regarding sensitivity of mixed-V_t benefits to initial designs and library options, we offer a "decision tree" that may help designers make implementation choices, as follows.

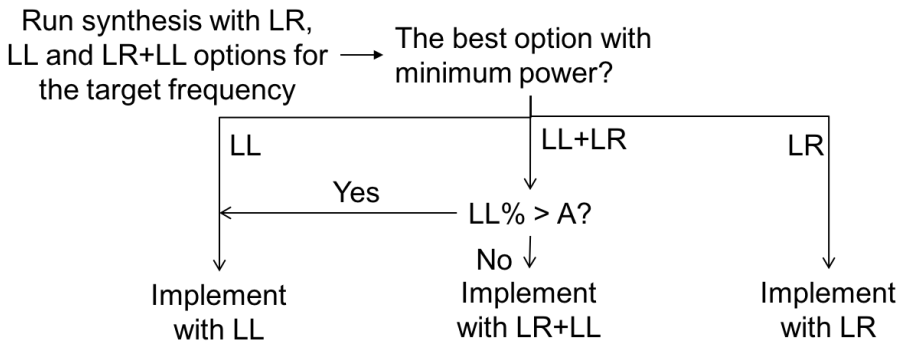


Figure 2.17: Notional decision tree for FDSOI implementation option choice.

Decision Tree

Figure 2.17 shows our notional 'decision tree", based on our experimental studies and observations, for implementation option choice in FDSOI. For the input RTL, logic synthesis results with LL, LR and mixed-V_t are needed to see which implementation option offers the minimum power for the target operating frequency. If mixed-V_t is the best option, measure the portion of LL cells (LL%) in the synthesized netlist. If LL% $> A$, it would be better to use LL option since we observe that mixed-V_t designs may not offer

better power if LL cells are dominant. For A, we empirically recommend to use 15%. This is because LL% is likely to increase in the presence of placement constraints (in our experimental results, we observe a typical increase of $\sim 3\times$). Further, with $LL\% >\sim 50\%$, not much power benefit is seen compared to pure LL designs (i.e., LL% = 100).

Difficulty of Fine-Grained Body-Biasing in FDSOI

Last, we would like to add a brief further discussion regarding the potential for fine-grained *body-biasing* in FDSOI, and an additional fundamental challenge for this optimization. We first state the SDP problem for the body-biasing context, as follows.

Problem formulation ("SDP-FBB") for forward body bias. Given an initial placed design implemented with LR cells only, we perform V_t swapping, sizing and placement optimization to define "LLFBB" (i.e., LL with FBB applied) regions under timing/placement constraints. In this problem formulation, we would only consider FBB on LL since the feasible range of FBB voltage on LR is very limited.

Input: A placed design, synthesized and optimized with LR cells.

Output: An optimized mixed-V_t netlist/placement, with FBB islands.

Constraints: The target operating frequency f_{fbb} at FBB mode should be X% higher than f_{nbb} (the operating frequency at zero/no body bias (NBB)). The Δ total power ($(p_{fbb} - p_{nbb})/p_{nbb}$) is no more than Y%, where p_{fbb} (resp. p_{nbb}) is the total power at FBB (resp. NBB) mode. LL/FBB regions should be rectilinear islands, and the area should be less than Z% of the total area of the design.

The SDP-FBB problem has a fundamental, *moving baseline* challenge inherent in setting the baseline for speed boost target. This is because both the baseline f_{nbb} and the target f_{fbb} change during cell-swapping optimization (i.e., LLFBB island generation), as illustrated in Figure 2.18. Thus, it is not straightforward to calculate a target frequency f_{fbb}. The moving baseline presents a chicken-egg situation: Once we generate an LLFBB island to improve timing by covering cells on the critical path, we can improve f_{fbb}. Meanwhile, f_{nbb} gets improved as well since cells in the LLFBB island automatically become LL. Eventually, an increased f_{nbb} sets a new target f_{fbb}, which induces a convergence issue.

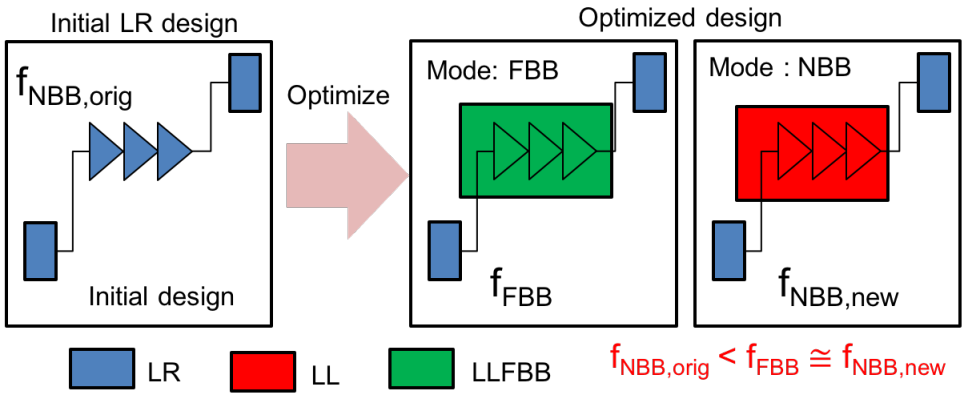


Figure 2.18: The "moving baseline" challenge: As f_{fbb} is improved, the value of f_{nbb} changes during the process of LLFBB island generation.

Looking Forward

Finally, we believe that future work must further elucidate the cost-benefit tradeoffs in fine-grain, mixed-V_t (and, body biasing-based) FDSOI. This will be essential to correct technology adoption decisions by product teams. Our work is only a first step toward this understanding. We believe that specific near-term research targets include identification of important parameters that determine SDP-friendly designs; consideration of clock distribution and on-chip variation in signoff analyses; inclusion of useful skew into the optimization flows; hold time considerations; and improved optimization heuristics for the SDP implementation problem.

2.4 Acknowledgments

Chapter 2 contains the reprint of Andrew B. Kahng and Hyein Lee, "Minimum Implant Area-Aware Gate Sizing and Placement", *Proc. ACM Great Lakes Symposium on Very Large Scale Integration*, 2014. Chapter 2 also contains the draft submitted to *Integration, the VLSI Journal*, Hamed Fatemi, Andrew B. Kahng, Hyein Lee, Jiajia Li and José Pineda de Gyvez, "Enhancing Sensitivity-Based Power Reduction for an Industry IC Design Context", 2018; the draft submitted to *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Hamed Fatemi, Andrew B. Kahng, Hyein Lee and José Pineda de Gyvez, "Heuristic Methods for Fine-Grain Exploitation of FDSOI", 2018. The dissertation author is a

main contributor to, and a primary author of, each of these papers.

I would like to thank my coauthors Hamed Fatemi, José Pineda de Gyvez, Andrew B. Kahng and Jiajia Li.

Chapter 3

Detailed Placement Optimizations for Advanced VLSI Technologies

Technology scaling to $10nm$ and below introduces complex intra-row and inter-row constraints in standard-cell detailed placement. Examples of such constraints are found in rules for drain-drain abutment, minimum implant region area and width, and oxide diffusion (OD) notching and jogging. In addition to placement rules, aggressive pitch scaling in sub-$10nm$ nodes has introduced complex routing rules which make detailed routing extremely challenging. Cell architectures have also been changed for better detailed routing. For example, metal layers below M1 are used to gain additional routing resources. New cell architectures wherein inter-row M1 routing is allowed force consideration of vertical alignment of cells. The layout complexities inherent in these new placement rules and new cell architectures motivate the introduction of a final legalization phase for standard-cell placement tools in advanced (particularly $10nm$ and $7nm$) foundry nodes.

This chapter presents two distinct methodologies for detailed placement optimization for advanced VLSI manufacturing. First, we develop a mixed integer-linear programming (MILP)-based placer, called **DFPlacer**, for final-phase design rule violation (DRV) fixing. DFPlacer finds (near-)DRV-free solutions considering various complex layout constraints including minimum implant width, drain-drain abutment, and oxide diffusion jogs. To overcome the runtime limitation of MILP-based approaches, we implement

a distributable optimization strategy based on partitioning of the block layout into windows of cells that can be independently legalized. Using layouts in an abstracted $7nm$ library, we find that DFPlacer fixes 99% of DRVs on average with minimal impacts on area and timing. We also study an area-DRV tradeoff between two types of standard-cell library strategies, namely, with and without dummy poly gates. Second, we propose a MILP-based detailed placement optimization to maximize direct vertical M1 routing utilization for congestion and wirelength reduction. Our optimization considers two new cell architectures in sub-$10nm$ nodes, i.e., *ClosedM1* which has 1D vertical M1 pins, and *OpenM1* which has M0 pins with more M1 routing resources. With our optimization, up to 6.4% (resp. 2.2%) total routed wirelength reductions and 14.4% (resp. 4.1%) #via12 reductions are achieved for *ClosedM1*-based (resp. *OpenM1*-based) designs, with no adverse timing impact.

3.1 Scalable Detailed Placement Legalization for Complex Sub-14nm Constraints

Continued technology scaling to the foundry $10nm$ node ($42nm$ minimum metal pitch, $36nm$ fin pitch) and below leads to more constraints in physical implementation. Not only do new metal-layer (back end of line, or BEOL) ground rules arise from multi-patterning techniques, but rules for device layers (front end of line, or FEOL) also become considerably more complex and restricted. For example, at the foundry $10nm$ node (henceforth referred to as **N10**), there are minimum width and area constraints for implant regions, as well as notch and jog width constraints for oxide diffusion (OD) regions. In older technology nodes, such layer rules were fairly benign: while of concern to the library cell designer, once the library cells were correctly designed, design rule violations (DRVs) could not occur during placement due to the correctness by construction of any non-overlapping cell placement.

Unfortunately, correctness by construction no longer holds for detailed placement at N10 and below. Cell sizes and minimum metal pitches have continued shrinking to stay on the Moore's Law density curve. However, patterning resolution in device (FEOL) layers has not kept pace due to challenges in device definition (e.g., ion implant) or lithographic variation (e.g., corner rounding). Thus, placing several 'legal' standard-cell layouts next to each other may cause violations of FEOL layer rules such as minimum

implant width or area [86] rules. Such violations could in theory be prevented with larger cell area budgets (similar in spirit to how BEOL colorability, especially on the M1 layer, can be preserved) that permit correct-by-construction (or, more precisely, "composable-by-construction") cell layout styles. However, this runs counter to a core purpose of shrinking to the next node, and reduces the return on investment from enabling that node. Our present work proposes a new, final phase of detailed cell placement that can potentially maintain placement legality in the face of new N10 FEOL rules – without loss of density, routability or performance metrics.

N10 FEOL and Cell Placement Constraints

Figure 3.1 illustrates the layout of an inverter cell in the N10 node. The figure shows two fins each for PMOS and NMOS.[19] Source nodes of PMOS and NMOS are connected to M2 power/ground rails with M1. The input ***A*** is connected to the PMOS and NMOS gates using middle-of-line (MOL), a complementary metal layer below M1 that is used for intra-cell routing. The output ***Y*** is connected to the drain nodes of PMOS and NMOS. The FEOL layers which affect legal placement (i.e., in the context of other cells' placements) include implant layer, oxide diffusion (OD) layer and poly, as follows.

- Implant layers, which indicate regions for ion implantation, decide the threshold (V_t) of transistors. Regions of the implant layer are typically aligned to the boundaries of standard cells.
- Oxide diffusion (OD) defines the active region of transistors.
- Dummy poly gates are inserted at the (vertical) standard cell boundaries to avoid edge device variability.

Minimum implant width (IW) constraints. Minimum implant width (IW) constraints induce placement illegalities due to both inter- and intra-row IW violations, as shown in Figure 3.2(a). Below, we refer to the inter-row IW violation as being of type IW1. We refer to the intra-row IW violation as being of type IW2. An example of IW1 is shown in the figure, where two same-V_t cells are misaligned vertically and thus result in a narrow, "staircase" implant layer shape. IW2 occurs when a narrow cell is

[19] A more typical library in N10 might have 9-track (M2 tracks) cell height, and three fins each for PMOS and NMOS, with a gear ratio of M2:fin pitch anywhere from 7:6 to 4:3.

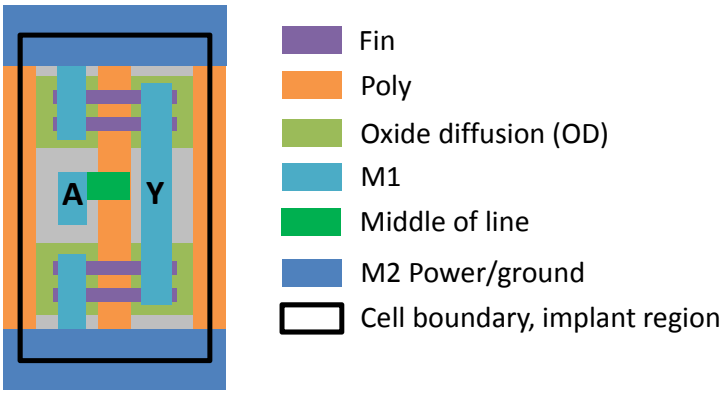


Figure 3.1: Illustration of inverter cell layout in N10 node.

sandwiched between different-V_t cells, which results in a narrow implant region. Interestingly, the IW rules cause interactions between placement and sizing optimizations (e.g., V_t-swapping) that compromise the notion of, e.g., "post-route leakage optimization". This interaction has been recently studied in [86].

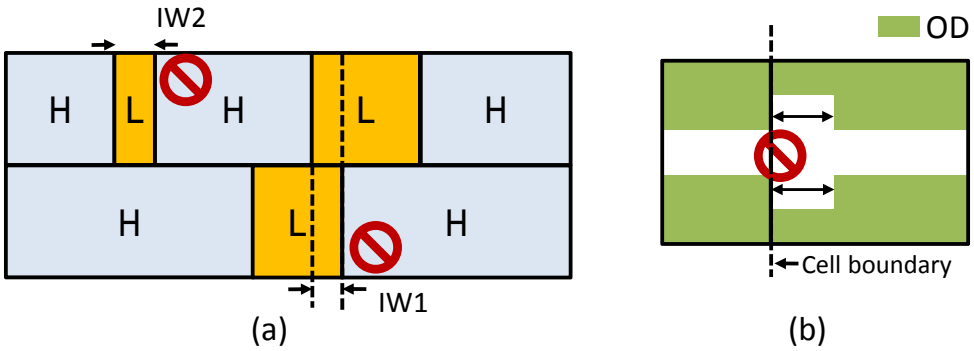


Figure 3.2: (a) Examples of minimum implant width violations [186]. (b) The design rule for OD jogs.

Minimum OD jog length (OW) constraints. Standard cells can have different oxide diffusion (OD) region heights according to functionality, drive strength, etc. When cells with different OD heights abut, OD jogs can result as shown in Figure 3.2(b). This is forbidden in N10 and below due to lithographic corner rounding and the consequent device performance variability, e.g., under misalignment. In N10, a minimum OD jog length rule is violated if the jog length is less than a given minimum value. Introducing sufficient spacing between the violating cells can cure the OD jog violation.

Drain-drain abutment (DDA) constraints. Dummy poly gates create extra dummy transistors connected to logic transistors within standard cells. The dummy transistors can induce leakage power and

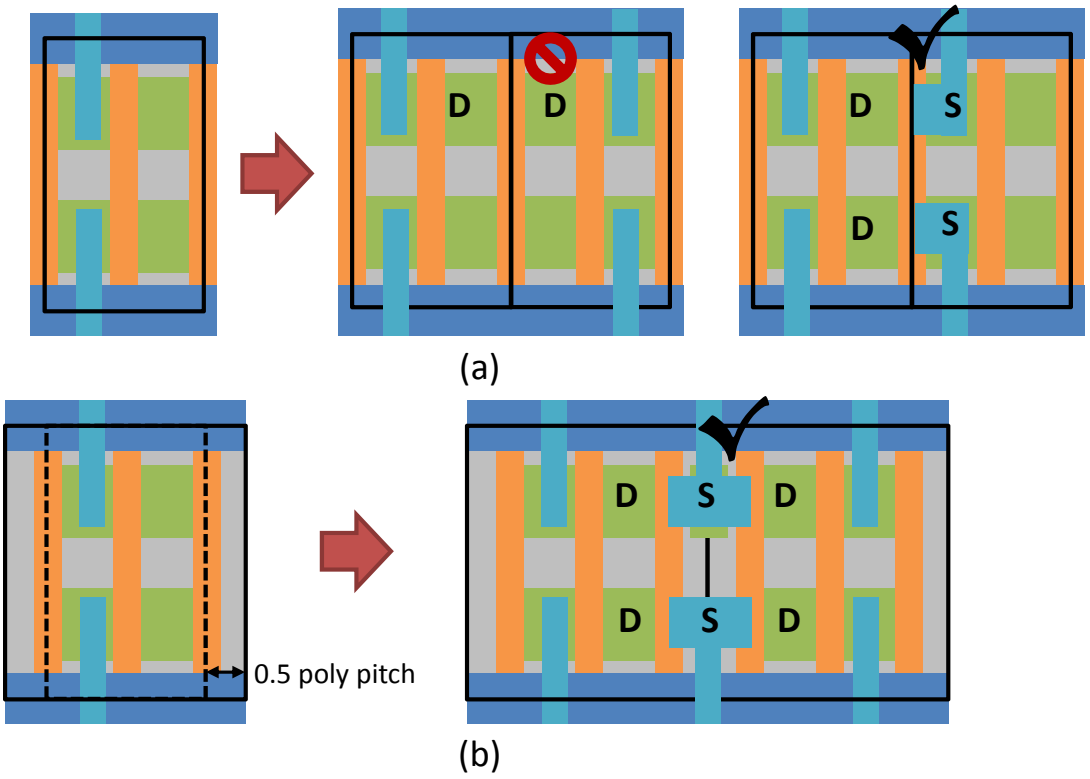


Figure 3.3: (a) Drain-drain abutment violation with an example standard cell layout. (b) Use of dummy poly gates in the library design style can avoid DDA violation in a correct-by-construction manner.

logic failure if they are not fully turned off. Hence, gate and source nodes of dummy transistors must be tied off to power/ground rails; in particular, if two drain nodes are abutted, an extra dummy poly gate is needed to create an additional source node to be tied up with power/ground rails. The recent work of Du and Wong [39] studies cell instance flipping as a way of mitigating this issue in detailed placement. Figure fig:d2d(a) depicts the DDA problem. The leftmost diagram shows an example inverter layout, and the middle and right diagrams respectively show drain-drain abutment (DDA) and no-DDA cases. To avoid the DDA problem, we can consider two approaches [2]: (i) a smart detailed placement with comprehension of DDA; and (ii) standard cells with embedded dummy poly gates as shown in Figure 3.3(b). With approach (ii), the width overhead for each cell is one poly pitch. (We study the area-DRV tradeoff between approaches (i) and (ii) in Section 3.1.3 below.)

This Work

As noted above, [86] and [39] have respectively made initial studies of IW- and DDA-induced placement issues. However, to our knowledge there is no existing work that addresses all the issues above simultaneously in detailed placement. Popular techniques used for conventional placement legalization, including graph-based, dynamic programming-based, etc. methods, appear ill-suited to handling of complex FEOL layer rules at N10 and below. For example, previous techniques have focused on removing overlaps between cells while maintaining the ordering of cells within a row, while minimizing half-perimeter wirelength or placement perturbation. Such previous works are largely single-row-based, and are applied row by row. Thus, they do not capture *inter-row* constraints such as IW1 that arise in N10. Furthermore, a number of implicit assumptions made by placement legalizers are broken when placement correctness by construction no longer holds, e.g., when more than two cells can interact and create DRVs. This challenges the use of dynamic programming frameworks, since decomposition into independent placement subproblems is no longer obvious. Finally, filler cell insertion has not previously been a concern of placement legalizers, but in N10 the filler cells can cause additional implant layer rule violations.

In this work, we propose a mixed integer-linear programming (MILP)-based placement legalization that considers complex N10 FEOL-layer design rules including minimum implant width, minimum oxide diffusion jogs and drain-drain abutment. We also propose a distributable optimization approach based on partitioning a given placement into many windows of cells, with each window being independently optimizable. The main contributions of our work are summarized as follows.

- We formulate as an MILP a placement problem that addresses new DRVs caused by complex N10 design rules. In contrast to previous approaches, our formulation captures new *inter-row* violation types. We further implement our solution approach in a prototype tool, **DFPlacer**.
- DFPlacer handles whitespace in the problem formulation and determines filler cell insertions to solve implant width constraint violations.
- We propose a distributable optimization based on partitioning of an input placement into windows of cells, and demonstrate that our optimization is scalable via this mechanism.

- We implement our proposed methods in C++ with OpenAccess 2.2.43 [193] and incorporate them into a commercial tool-based placement and routing (P&R) flow for evaluation.
- A further study provides insight into timing and area impacts of the dummy poly gate library cell strategy, using two kinds of libraries: (i) standard cells with dummy poly gates (drain-drain abutment violation-free) and (ii) standard cells without dummy poly gates.

3.1.1 Related Work

We now summarize relevant previous works on detailed placement and placement legalization.

Dynamic programming-based approaches. Dynamic programming (DP), typically for a single cell row, has been used by a number of authors. Kahng et al. [89] use DP to legalize placement of a single row with various minimization objectives: total perturbation, maximum perturbation, and wirelength. A shortest-path algorithm is applied to a directed acyclic graph constructed from the input ordering of cells. Gupta et al. [56] perform detailed placement optimization to enable sub-resolution assist feature insertion for improved manufacturability. A DP-based single row placement achieves this *assist-feature correctness* (AFCorr) while minimizing (timing criticality-weighted) perturbations of cell locations. Subsequent work addresses a 2D formulation that considers both horizontal and vertical interactions between adjacent cells [57]. The *2D AFCorr* approach uses DP in which vertical and horizontal costs are calculated with restricted perturbations. Hur and Lillis [70] propose *optimal interleaving* for intra-row optimization in detailed placement. Their work splits the cells of a single row into two groups with a given window size, and the two sequences are optimally interleaved via DP while preserving the initial relative ordering of cells in each group. At the global placement level, cells are assigned to bins and optimized via *relaxation-based local search*.

Integer Linear Programming (ILP)-based approaches. Another important class of previous methods is based on integer linear programming. Ramachandaran et al. [132] apply branch-and-price for improved scaling of the placement optimization. Li and Koh [100] propose ILP-based detailed placement approaches using *placement site variables*. Dantzig-Wolfe decomposition is applied to improve scalability, and single-cell-placement (SCP) variables enable grouping and mapping of placement site variables into

patterns. The extension [101] supports mixed-size circuits and improves runtime by bounding solution spaces. In our present work, we begin with the MILP model of [100] [101], extending it to provide the first-ever comprehensive support (to our knowledge) of N10-relevant design rules such as minimum implant width, diffusion jogs and drain-drain abutment.

N10 design rules-aware placement. Du and Wong [39] address the abutment of source and drain in FinFET-based cell placement (i.e., for the foundry $14nm$ node onward), where the DDA constraint becomes prominent. The authors use cell flipping and adjacent-cell swapping as underlying operations for detailed placement perturbation that minimizes drain-drain abutments. As in [89], the authors of [39] apply a shortest-path algorithm with their proposed graph model, in which each operation and the violations are modeled as nodes and node/edge costs, respectively. However, the approach only swaps and flips cells within a single row, and does not handle interactions between placement rows. Hence, the optimization is made with respect to a highly restricted portion of the overall detailed placement solution space. Moreover, DDA-related optimization cannot be performed in isolation at the N10 node: many other neighborhood-related constraints (e.g., implant, oxide diffusion (OD), etc.) have interactions with, and constrain, the drain-drain abutment solution. In our present work, we handle neighborhood-related constraints along with drain-drain abutment, with a larger solution space that includes multiple rows.

3.1.2 Our Approach

Problem Formulation

We now formulate a MILP for our detailed placement problem to address N10-related design rules including IW, DDA and OW in Section 3.1. Our notation is described in Table 3.1.

Table 3.1: Notations.

Notation	Meaning
C, R, Q	sets of cells, rows, columns
f_c	a binary indicator of whether cell c is flipped
$x(y)_{c,init}$	initial x (y) coordinate of cell c
s_{crq}	a binary indicator of whether cell c occupies site (r, q)
K_c	a set of candidate states of cell c
λ_c^k	a binary indicator of the k^{th} candidate state for cell c
x_c^k, y_c^k	x and y coordinates corresponding to λ_c^k
f_c^k	f_c corresponding to λ_c^k
s_{crq}^k	s_{crq} corresponding to λ_c^k
m_{rq}	inter-row variable for IW1
h_{rq}	intra-row variable for IW2
W	minimum implant width (unit: site)

$$\textbf{Minimize:} \quad \sum_{c \in C} (|x_c - x_{c,init}| + |y_c - y_{c,init}|) \tag{3.1}$$

Subject to:

$$\sum_{k \in K_c} \lambda_c^k = 1, \qquad \forall c \in C, \;\; \lambda_c^k \in \{0, 1\} \tag{3.2}$$

$$f_c = \sum_{k \in K_c} f_c^k \lambda_c^k \tag{3.3}$$

$$x_c = \sum_{k \in K_c} x_c^k \lambda_c^k, \;\; y_c = \sum_{k \in K_c} y_c^k \lambda_c^k, \;\; \forall c \in C \tag{3.4}$$

$$s_{crq} = \sum_{k \in K_c} s_{crq}^k \lambda_c^k, \quad \forall c \in C \tag{3.5}$$

$$\sum_{c \in C} s_{crq} \leq 1, \qquad \forall q \in Q, r \in R \tag{3.6}$$

For a given input layout, our objective is to minimize the sum of cell displacements while achieving a legal placement with respect to given N10 design rules. We assume a given perturbation range for each cell g ($g.l$, $g.r$, $g.t$ and $g.b$ are the maximum allowed displacements of the cell in the left, right, top and bottom directions, respectively); a cell cannot move beyond its given perturbation range. Thus, we have a limited number of possible states (locations and orientations) within g, for each cell. To represent each

candidate state for a cell, we adopt the single-cell-placement (SCP) model of [101]. The binary SCP variable λ_c^k represents a candidate state k for a cell c. The variable λ_c^k is associated with the location and orientation of cell c, e.g., x_c^k, y_c^k and f_c^k, which are pre-defined values. Also, s_{crq}^k, where $r \in |R|$ and $q \in |Q|$, is pre-defined for λ_c^k.

From Constraint (3.2), exactly one state is chosen for cell c among multiple candidate states $\lambda_c^k, k \in K_c$; this determines the location and orientation of c. Constraints (3.3), (3.4) and (3.5) determine the final x, y of cell c and s_{crq} for $r \in |R|, q \in |Q|$ from a selected candidate site for cell c. To ensure a legal placement (no overlap), Constraint (3.6) forces a site at (r, q) to be occupied by at most one cell. In addition to the basic formulation, we add extra constraints to address OW, DDA, IW1 and IW2 rules, as follows.

OW and DDA constraints. To handle OW and DDA constraints, we pre-characterize all adjacency conditions which violate OW and/or DDA for each library cell pair. We note that our pre-characterization considers the orientations of cells (i.e., the adjacency conditions change depending on the orientations of cells). We then generate a set P of forbidden pairs of λ_c^k. Based on P, we formulate Constraint (3.7) for every forbidden pair $(\lambda_{c_1}^i, \lambda_{c_2}^j)$.

$$\lambda_{c_1}^i + \lambda_{c_2}^j \leq 1 \qquad \text{where } c_1, c_2 \in C, (\lambda_{c_1}^i, \lambda_{c_2}^j) \in P \tag{3.7}$$

IW1 and IW2 constraints. IW1 violations occur across rows when vertically-adjacent same-V_t layers form a narrow staircase shape with width less than the minimum implant width (see Figure 3.2(a)). To handle IW1, we define a 0-1 *inter-row variable,* m_{rq}, that indicates whether the site at (r, q) (row r and column q) and the site at $(r + 1, q)$ have the same V_t ($m_{rq} = 1$) or not ($m_{rq} = 0$). Figure 3.4(a) illustrates the m_{rq} variables and IW1 constraints. As shown in the figure, if a 0-1 sequence of m values is found (e.g., m_{12}, m_{13}), the implant region has a staircase shape, and hence $(W - 1)$ consecutive m variables must be one. Thus, we formulate constraints that, if $m_{r(q-1)} = 0$ and $m_{rq} = 1$, force at least W consecutive inter-row variables $m_{rq} = \ldots = m_{r(q+W-1)} = 1$, so as to satisfy IW1 (e.g., m_{13}, m_{14}, $m_{15} = 1$ where $W = 3$, in Figure 3.4(a)).

IW2 violations occur when small-width cells are sandwiched in between different-V_t cells in the same row. Similar to how we handle IW1, we define a 0-1 *intra-row variable*, h_{rq}, that indicates whether the site at (r, q) and the site at $(r, q+1)$ have the same V_t ($h_{rq} = 1$) or not ($h_{rq} = 0$), as shown in Figure 3.4(b). If $h_{rq} = 0$, i.e., sites (r, q) and $(r, q+1)$ have different V_t, we force $(W-1)$ consecutive binary variables $h_{r(q+1)} = \ldots = h_{r(q+W-1)} = 1$, so as to have at least W consecutive same-V_t sites. Figure 3.4(b) shows the case when $h_{rq} = 0$, where $r = 1$, $q = 2$, $W = 3$.

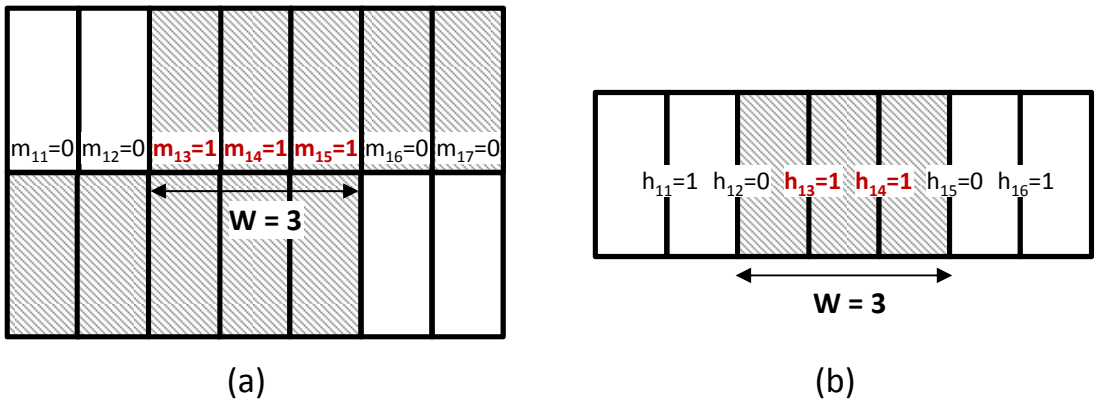


Figure 3.4: (a) Inter-row variable m_{rq} for IW1. (b) Intra-row variable h_{rq} for IW2. The color (gray and white) of regions indicates V_t.

The generalized constraints for IW1 and IW2 are as follows:

$$m_{r0} = 0, h_{r0} = 0 \qquad 1 \le r < |R| \tag{3.8}$$

$$m_{rq} + (1 - m_{r(q+1)}) + y_{r(q+2)} \ge 1$$
$$0 \le q < |Q| - W,\ 1 \le r < |R| \tag{3.9}$$

$$y_{rq} \le m_{r(q+w)}$$
$$2 \le q \le |Q| - 2,\ 1 \le r < |R|,\ 0 \le w < W - 1 \tag{3.10}$$

$$h_{rq} + z_{rq} \ge 1$$
$$0 \le q < |Q| - W,\ 1 \le r < |R| \tag{3.11}$$

$$z_{rq} \le h_{r(q+1+w)}$$
$$2 \le q \le |Q| - 2,\ 1 \le r < |R|,\ 0 \le w < W - 1 \tag{3.12}$$

- Constraint (3.8) initializes the leftmost m and h variables where $q = 0$.
- Constraint (3.9) detects the condition of $m_{rq} = 0$ and $m_{r(q+1)} = 1$, and forces $y_{r(q+2)} = 1$.
- When $y_{r(q+2)} = 1$, Constraint (3.10) forces $(W-1)$ consecutive binary variables $m_{r(q+2)} = \ldots = m_{r(q+2-(W-1))} = 1$.
- Constraint (3.11) detects the condition of $h_{rq} = 0$ and forces $z_{rq} = 1$.
- When $z_{rq} = 1$, Constraint (3.12) forces $(W-1)$ consecutive binary variables $h_{r(q+1)} = \ldots = h_{r(q+(W-1))} = 1$.

We now describe our method of obtaining inter- and intra-row variables (m_{rq} and h_{rq}). We first set the V_t of cell c as the binary vector $\vec{k}_c$. The length of $\vec{k}_c$ is determined by $\lceil log_2(n_{V_t} + 1) \rceil$ where n_{V_t} is the number of available V_t options. For example, if we have three V_t options, then $\vec{k}_c$ is $\{k_c^1\ k_c^2\}$. Concretely, the binary vectors $\{0\ 1\}$, $\{1\ 0\}$ and $\{1\ 1\}$ represent HVT, NVT and LVT, respectively. We then define the *V_t variable* $\vec{v}_{rq}$ as a binary vector variable $\{v_{rq}^1\ v_{rq}^2\}$ indicating the V_t of the site (r, q). Given that $m_{rq} = 1$ if $\vec{v}_{rq} = \vec{v}_{(r+1)q}$, we add the following constraint to obtain m_{rq}:

$$m_{rq} = \overline{(v_{rq}^1 \oplus v_{(r+1)q}^1) + (v_{rq}^2 \oplus v_{(r+1)q}^2)} \tag{3.13}$$

Constraint (3.13) is rewritten in our MILP formulation, using binary variables u_1, u_2 and $\overline{m_{rq}}$, as follows:

$$\begin{aligned}
&\overline{m_{rq}} + m_{rq} \leq 1; \\
&\overline{m_{rq}} \leq u_1 + u_2;\ \overline{m_{rq}} \geq u_1;\ \overline{m_{rq}} \geq u_2; \\
u_1 \leq v_{rq}^1 + v_{(r+1)q}^1;&\ u_1 \geq v_{rq}^1 - v_{(r+1)q}^1; \\
u_1 \geq v_{(r+1)q}^1 - v_{rq}^1;&\ u_1 \leq 2 - v_{rq}^1 - v_{(r+1)q}^1 \\
u_2 \leq v_{rq}^2 + v_{(r+1)q}^2;&\ u_2 \geq v_{rq}^2 - v_{(r+1)q}^2; \\
u_2 \geq v_{(r+1)q}^2 - v_{rq}^2;&\ u_2 \leq 2 - v_{rq}^2 - v_{(r+1)q}^2
\end{aligned} \tag{3.14}$$

Similarly, h_{rq} can be formulated as follows:

$$h_{rq} = \overline{(v^1_{rq} \oplus v^1_{r(q+1)}) + (v^2_{rq} \oplus v^2_{r(q+1)})} \tag{3.15}$$

We also consider whitespace (empty sites), which can be filled with filler cells. We have the freedom to choose V_t of filler cells to satisfy IW1 and IW2 constraints. To exploit this flexibility, we define a binary vector variable $\vec{e}_{rq}$= $\{e^1_{rq}e^2_{rq}\}$ which indicates V_t of the site at (r, q). From variables $\vec{k}_c$, s_{crq} and $\vec{e}_{rq}$, the binary vector variable $\vec{v}_{rq}$ is defined as follows:

$$\vec{v}_{rq} = \sum_{c \in C} \vec{k}_c \cdot s_{crq} + \vec{e}_{rq} \tag{3.16}$$

Thus, $\vec{v}_{rq}$ is determined by either $\sum_{c \in C} \vec{k}_c \cdot s_{crq}$ or $\vec{e}_{rq}$. We add a constraint below for $\vec{e}_{rq}$:

$$e^1_{rq} \leq 1 - \sum_{c \in C} s_{crq}; \quad e^2_{rq} \leq 1 - \sum_{c \in C} s_{crq} \tag{3.17}$$

Constraint (3.17) states that if a site is occupied by any cell, $\vec{e}_{rq}$= 0. Then, Constraint (3.16) becomes independent of $\vec{e}_{rq}$. Otherwise, Constraint (3.16) becomes $\vec{v}_{rq} = \vec{e}_{rq}$.

Analysis of the number of variables and constraints. The number of variables and constraints depends on the number of sites in a target window ($|R| \cdot |Q|$), the number of instances ($|C|$) and the size of the perturbation range ($g.size = (g.l + g.r) \cdot (g.t + g.b)$).

- The number of variables s_{crq} is $|C| \cdot |R| \cdot |Q|$.
- The number of variables x_c, y_c is (each) $|C|$; the number of variables $x^k_c, y^k_c, \lambda^k_c$ is (each) $g.size \cdot |C|$.
- The number of inter-/intra-row variables m, h is (each) $|R| \cdot |Q|$.

- The number of variables v and e is (each) $n \cdot |R| \cdot |Q|$, where n is $\lceil log_2(n_{V_t} + 1) \rceil$.
- The numbers of Constraints (3.2), (3.4), (3.5) and (3.6) are $|C|$, $|C|$, $|C| \cdot |R| \cdot |Q|$ and $|R| \cdot |Q|$, respectively.
- The number of Constraint (3.7) is $g.size \cdot |C|^2$.
- The number of Constraints (3.9), (3.10), (3.11) and (3.12) is (each) $|R| \cdot |Q|$.

Overall Flow

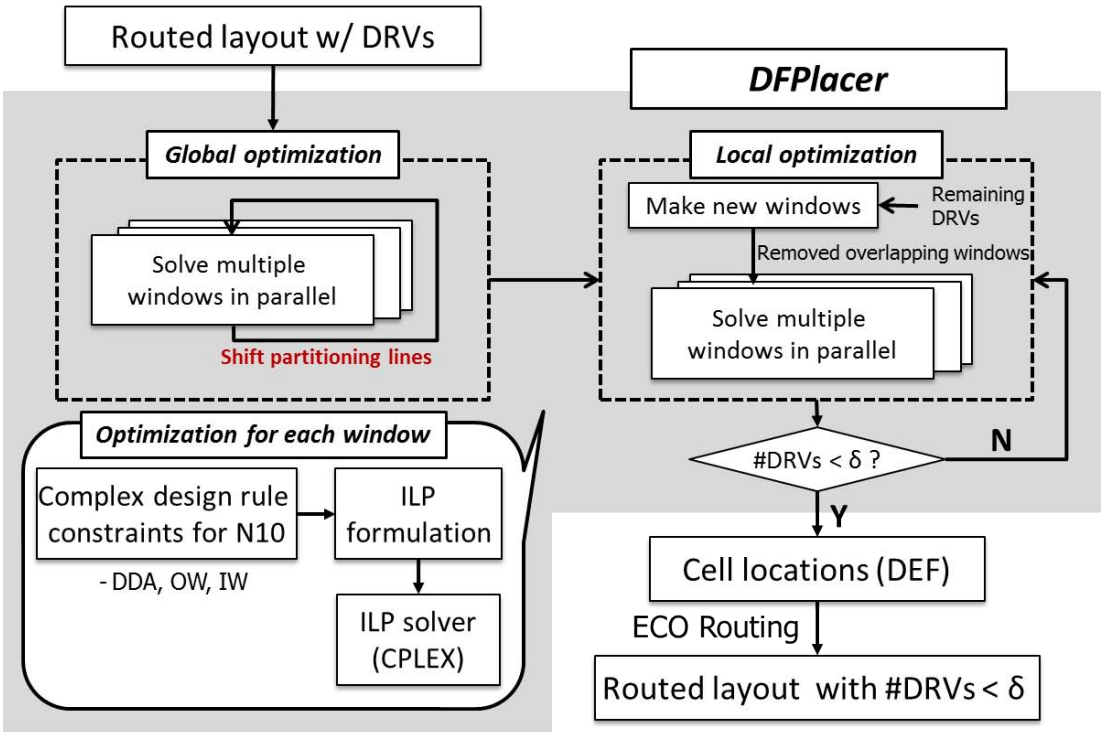


Figure 3.5: Overall flow of detailed placement legalization.

We implement our flow in C++ with *OpenAccess 2.2.43* [193] to support LEF/DEF [186], and with *IBM ILOG CPLEX Optimization Studio v12.5.1* [180] as our MILP solver. Figure 3.5 shows the overall flow of our tool, which we call DFPlacer. DFPlacer has two optimization stages: global and local optimization. In the global optimization, we split the given routed layout T uniformly into a set of windows D and optimize each of the windows $d \in D$ in parallel. We use a fixed boundary margin b for each window

to enable independent optimization among windows. In the local optimization, we generate a new window for each remaining violation $\gamma \in \Gamma$ such that the violation is located at the center of the window. We then remove overlapping windows so that no window affects another. With the new set of windows D', we optimize each window $d' \in D'$ again in parallel. The output cell location solution is saved in DEF file format, which can be fed into a commercial P&R tool. We perform ECO routing with the solution and finally obtain a new layout T_{opt} with number of violations $|\Gamma|$ less than the given target number δ.

Algorithm 10 Overall flow of DFPlacer.

Procedure $DFPlacer(T, U, z, b, g, \delta)$
Input : Layout T, set of design rule constraints U, window size z, boundary margin b, perturbation range g, target number of DRVs δ
Output : Layout T_{opt} with $|\Gamma| < \delta$

```
1:  // Global optimization
2:  for i = 1 to 3 do
3:      A set of windows D ← Partition(T, i, z, b, g);
4:      Solve all MILP instances for windows D in parallel;
5:      Update MILP solutions to T;
6:  end for
7:  // Local optimization
8:  Γ ← GetDRV(T, U);
9:  while |Γ| < δ do
10:     D ← ∅;
11:     for all γ ∈ Γ do
12:         d ← MakeNewWindows(T, γ, z, b, g);
13:         D ← D ∪ d;
14:     end for
15:     D' ← NonOverlapWindows(D)
16:     Solve all MILP instances for windows D' in parallel;
17:     Update MILP solutions to T;
18:     Γ ← GetDRV(T, U);
19:     if |Γ| is the same as |Γ| in the previous iteration then
20:         IncreaseWindow(z);
21:         IncreasePerturb(g);
22:     end if
23: end while
24: T_opt ← T;
25: return T_opt;
```

Algorithm 10 gives further details of our optimization flow. In Lines 2-6, the global optimization phase solves D in parallel with a small perturbation range g (e.g., $g.l = 4$, $g.r = 4$, $g.t = 1$ and $g.b = 1$

sites) of cells. This distributable method overcomes the runtime limitation of MILP-based approaches and fixes more than 90% of initial $|\Gamma|$ (see Figure 3.7). In Line 3, we first partition a given routed T into D, and we solve each $d \in D$ in parallel using *OpenMP* [192] in Line 4. We set a window width $z.w$ as 47 sites, and a window height $z.h$ as nine cell rows in our experiments.[20] When running optimizations for the windows in parallel, we set the vertical (resp. horizontal) boundary margin $b.v$ (resp. $b.h$) so that the solution of one window can be isolated from the solutions of neighbor windows. We set $b.v$ as the minimum implant width W and $b.h$ as two cell row heights. Figure 3.6 shows the boundary margin in green color. We then update the MILP solutions to the layout T.

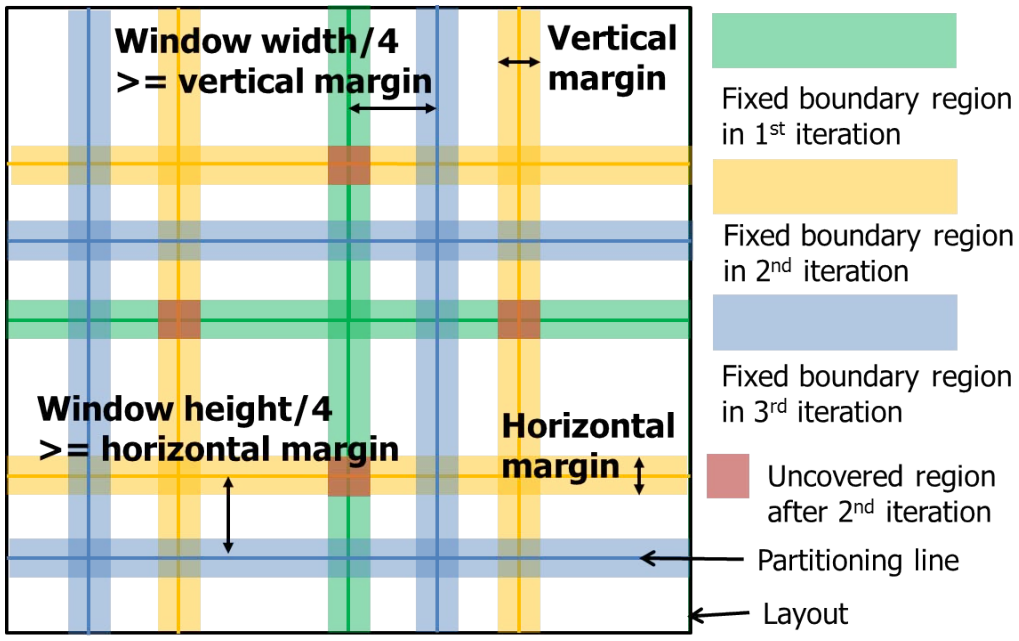


Figure 3.6: Partitioning of layout for parallel global optimization.

Since the fixed boundary cells corresponding to b of the first iteration can contain DRVs which are not fixed in the first iteration, we perform a second iteration with a new partitioning that is shifted by half of $z.w$ and $z.h$ in the x- and y-directions, respectively; these are shown in yellow color in Figure 3.6. We then partition the current T into a new D and solve the corresponding MILP instances to fix the violations

[20] Window size affects the tradeoff between the number of remaining violations $|\Gamma|$ after global optimization and the runtime of global optimization. Our studies of different window sizes (i.e., $z.w$ ranging from 40 sites to 55 sites and $z.h$ ranging from five cell row heights to 11 cell row heights) find that for a sample design (JPEG) a width of 47 sites and a height of nine cell row heights empirically achieves a good outcome ($< 10\%$ of initial $|\Gamma|$) with relatively small runtime (< 30 minutes). We therefore use this window size in all of our reported experiments.

Γ remaining from the first iteration. We then update the MILP solutions to T. Even after the first and second iterations, DRVs could still exist in the intersections of the fixed boundary regions (red color in Figure 3.6). To fix the Γ in the uncovered intersection region, we perform a third iteration that has new partitioning lines shifted by a quarter of $z.w$ and $z.h$ in x-direction and y-direction. Note that a quarter of $z.w$ and of $z.h$ should respectively be larger than or equal to $b.v$ and $b.h$. This ensures that the windows of the third iteration contain the uncovered regions, such that the fixed boundary region of the third iteration is not overlapped with the uncovered intersection region.

The small window size and perturbation range used in global optimization restricts the solution space, potentially leading to infeasible solutions for certain windows. To fix the remaining Γ, we perform the local optimization in Lines 8-24. For each DRV, the function $MakeNewWindows()$ creates a new window whose center is the DRV point. In Line 15, $NonOverlapWindows()$ picks a set of disjoint windows D' to process in parallel. We then update the solutions to the current T and check the remaining Γ (Lines 16-17). In Lines 19-22, if the current $|\Gamma|$ is the same as the $|\Gamma|$ of the previous iteration, we increase $z.w$ by 10 sites and $z.h$ by one cell height. For perturbation range, we increase $g.l$, $g.r$, $g.t$ and $g.b$ by 2, 2, 1 and 1 sites, respectively. When the current $|\Gamma|$ is less than the target number of DRVs δ, we save the current T as T_{opt} and terminate the optimization. In our experiment, we set δ as 1% of initial $|\Gamma|$.

3.1.3 Experimental Setup and Results

Experimental Setup

We evaluate *DFPlacer* using two open-source designs (AES_*, JPEG_*) [191], an ARM Cortex M0 without memories (M0_*) and a 3×ARM Cortex M0 without memories (M0x3_*). We synthesize these testcases from RTL, and perform P&R with an abstracted $7nm$ dual V_t library. Our RTL-to-layout flow uses *Synopsys Design Compiler H-2013.03-SP3* [195] and *Cadence Encounter Digital Implementation System 13.1* [171] for logic synthesis and P&R, respectively. All experiments are performed with 40 threads on a $2.6GHz$ Intel Xeon E5-2690 dual-CPU server. In principle, the number of threads could be as large as the number of layout windows.

Table 3.2: Testcases used in the experiments.

Design	#Inst	LVT (%)	Util. (%)	WL (μm)	Area (μm^2)	WSS (ps)	WHS (ps)
M0_nd	8260	52	77	114685	7668	38	0
M0x3_nd	27248	56	80	392540	24463	126	0
JPEG_nd	47948	51	77	694624	49629	12	0
M0_d	8238	51	77	116866	8668	93	1
AES_d	12491	54	80	150632	10596	58	0
M0x3_d	26690	55	79	409579	27400	107	0
JPEG_d	48317	52	77	764738	55824	13	0

Libraries and design rules. We use a prototype $7nm$ standard-cell library from a leading IP provider. Since our design enablement for the $7nm$ technology is missing detailed BEOL technology information such as RC values and BEOL stack options, we scale the library to use a $28nm$ BEOL stack, following the methodology described in [60]. The site width and height are $0.136\mu m$ and $0.9\mu m$; these values correspond to placement site and cell row height parameters of the $28nm$ enablement. For design rules, we set the OW, IW1 and IW2 rules as four site widths. To check for DDA and OW violations, we pre-characterize all pairs of standard cells in the $7nm$ library. The library has 62 standard cells and the total number of pairs is 15376 (= 62×62×2×2), including cell flipping. For the standard cells without dummy poly gate, 7172 pairs out of the 15376 pairs violate the DDA constraint, and these pairs require at least one site space. Similarly, with the 4 site widths for OW, 280 out of the 15376 pairs violate the OW constraint, and such pairs also require one site space.[21]

Tradeoff between area/wirelength and DRVs. Table 3.2 shows the testcases used in our experiments. LVT, WSS, WHS and WL respectively indicate the portion of LVT cells, the worst setup and hold slacks, and wirelength. We assign V_t to cells uniformly to create more IW1 and IW2 violations. We use two kinds of libraries: (i) without dummy poly gates (CWOD) and (ii) with dummy poly gates (CWD). CWD is designed with dummy poly gates inserted to avoid interactions between cells which create DDA violations. For the CWD library, cell width is increased by one poly pitch compared to the CWOD library. The suffixes *_*d* and *_*nd* indicate that the designs are implemented with CWD and CWOD libraries, respectively. The same initial netlists are used for both *_*d* and *_*nd*. While comparing *_*d* and *_*nd*

[21] Based on our OW rule and library, all pairs of standard cells that violate OW constraints require only one site space. However, depending on the OW rule and library, some pairs of standard cells could require two or more site spaces.

designs, we observe that the average wirelength and area overhead of designs implemented using libraries with dummy poly gates are 7% and 14%, respectively. In terms of DRVs, *_nd* testcases have 134%∼176% more initial DRVs as reported in the second and fourth columns of Table 3.3.

Experimental Results

Table 3.3: Results of DFPlacer.

Design	IW #Vio.		DDA/OW #Vio.		WSS (ps)		WHS (ps)		ΔWL (%)	Max. Δloc. (μm)	Avg. Δloc. (μm)	#Changed cells (%)	CPU total (sec)	
	Init	Final	Init	Final	Init	Final	Init	Final					Global	Total
M0_nd	926	11	1611	14	38	83	0	0	2.79	2.89	0.56	4489 (54%)	768	2820
AES_nd	1771	16	1900	18	90	71	0	-1	3.42	2.89	0.52	5939 (49%)	787	2992
M0x3_nd	3514	17	4230	48	126	113	0	0	2.90	3.02	0.51	12752 (47%)	957	6897
JPEG_nd	4056	29	12024	135	12	22	0	0	2.30	8.99	0.70	24169 (50%)	1788	11983
M0_d	988	10	0	0	93	85	1	0	3.04	2.89	0.57	2996 (36%)	161	434
AES_d	1566	11	0	0	58	80	0	0	3.10	2.89	0.54	3852 (31%)	425	1207
M0x3_d	2810	27	0	0	107	105	0	-2	2.14	2.89	0.58	9340 (35%)	517	1336
JPEG_d	6296	43	0	0	13	81	0	0	-0.57	3.02	0.49	12244 (27%)	954	1401

Table 3.3 summarizes the number of DRVs (#Vio.), WSS, WHS, ΔWL, maximum Δlocation (Max. Δloc.), average Δlocation (Avg. Δloc.), the number of moved cells (#Changed cells) and runtime. Our DFPlacer fixes more than 99% of initial violations in runtime that is reasonable for practical contexts. From a timing perspective, ΔWSS (i.e., final WSS − init WSS) ranges from -19ps to 68ps, but all final designs have no negative WSS. Similar to WSS, ΔWHS ranges from -2ps to 0ps. The timing impact is small since most of the cells are moved within a given small perturbation range. Some cells can be moved more than 20 sites (i.e., $0.136 \times 20 = 2.72\mu m$) from their initial locations due to the accumulated displacement in the iterative local optimization. However, those cells are less likely to be in the most critical path, which is how the WSS or WHS would worsen. Also, the positive ΔWSS implies that there is room to improve timing, and that we could potentially co-optimize the timing along with DRV fixing in detailed placement legalization. This is a direction of ongoing work.

On the other hand, DFPlacer increases wirelength by up to 3%. The accumulated displacements of cells and the limited pin access for the standard cells in N10 could be causes of this wirelength increase. Between *_nd and *_d testcases, the ΔWL% of *_nd cases is similar or slightly larger. We believe that this is because IW violations are harder to fix compared to the OW and DDA violations, since the constraints are more complex. The rate at which the number of IW violations reduces is slower than that for OW and DDA violations. Also, since the CWD library cells are larger than the CWOD library cells, the

displacement of cells in *_d cases might have more impact on the wirelength increase. Therefore, % WL increase of *_d cases is smaller in general, but not necessarily always less than that of corresponding *_nd cases.

Columns *Max.* $\Delta loc.$ and *Avg.* $\Delta loc.$ show maximum and average cell displacement, respectively. We observe that the average cell displacement for all designs is up to $0.70\mu m$, which is $\sim$5 sites' width. The maximum displacement is up to $8.99\mu m$ for JPEG_nd. For other designs, the maximum displacement is similar to the half-perimeter of the perturbation range used in the global optimization ($2.888 = 0.9 \cdot 2 + 0.136 \cdot 8$ microns).

When we compare the results of designs with CWOD and CWD, we see a tradeoff between area and the number of DRVs (and runtime). We observe that the area overhead of using cells with dummy poly gate is 14% on average (up to 19%). However, the number of DRVs decreases by 61% on average (up to 64%). This affects the runtime of detailed placement legalization.

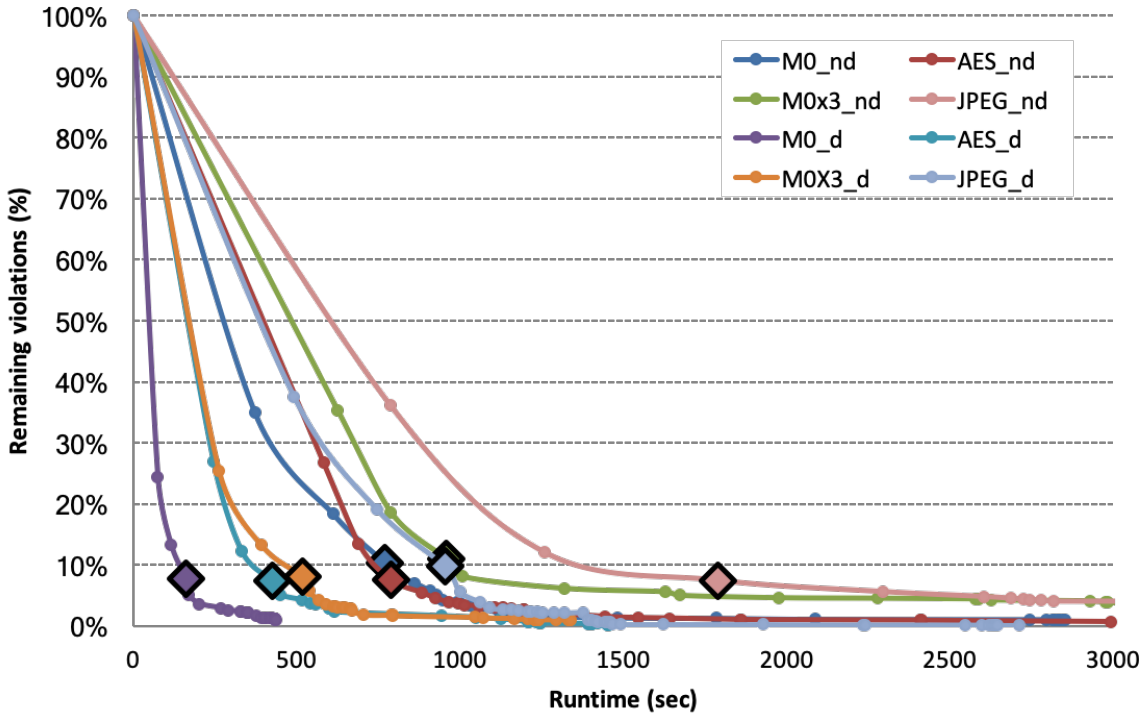


Figure 3.7: Remaining violations versus runtime. Each dot indicates an iteration; after the third iteration, local optimization is performed. The diamond-shaped markers represent third-iteration points.

Figure 3.7 shows the remaining number of DRVs (%) versus runtime (sec). Each dot stands for an iteration of the optimization, and the third iteration points are marked with diamond-shaped markers. During the global optimization, which includes first, second and third iterations, the remaining violations drop quickly; ∼90% of DRVs are fixed in most of the designs during the global optimization. The runtime of the global optimization phase still increases with the problem size. However, with added computing resources to run windows of cells in parallel, the runtime can be further reduced. After the third iteration, when entering into local optimizations, the rate of decrease of the number of DRVs becomes much lower, implying that DFPlacer spends considerable time to fix the last few DRVs. This is because these last DRVs cannot be solved with small window sizes and perturbation ranges in the global optimization; thus, DFPlacer tries to resolve them in the local optimization by increasing window sizes and perturbation ranges. The poor scaling of MILP solution versus instance size leads to the observed runtimes.

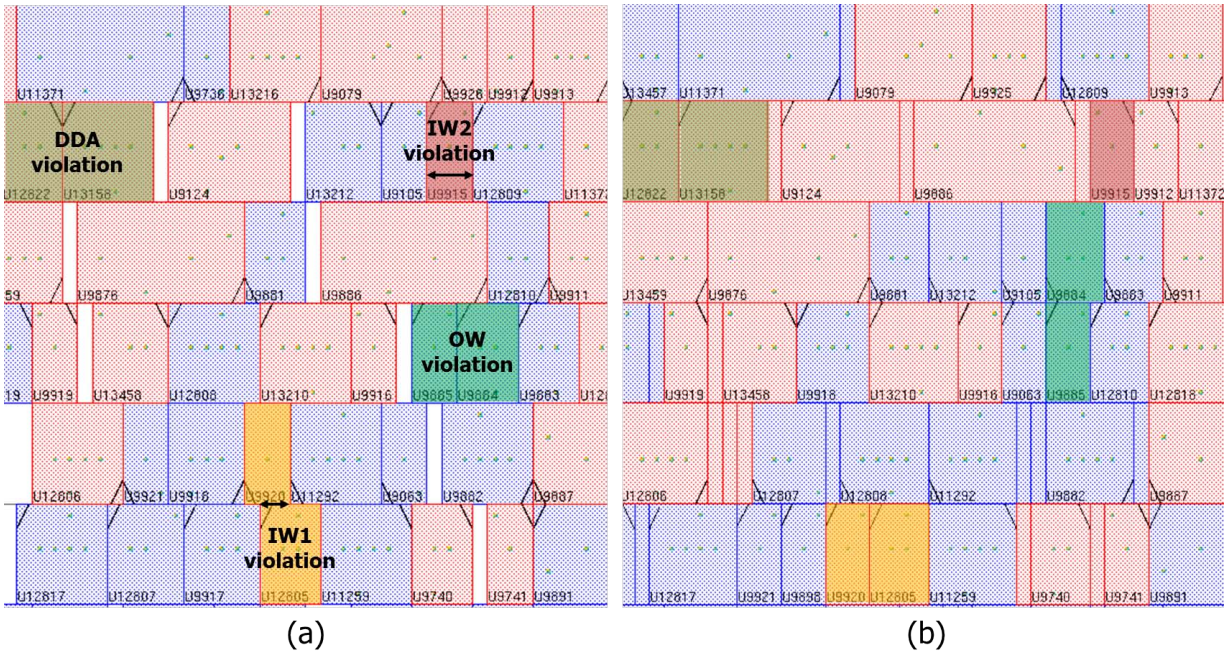


Figure 3.8: (a) Layout before optimization. OW, DDA, IW1 and IW2 violations are respectively highlighted in green, light green, yellow and brown colors. (b) Layout after optimization. The DRVs in the initial layout are fixed.

Figures 3.8(a) and 3.8(b) respectively show layout snapshots from the pre- and post-detailed placement legalization phases. In Figure 3.8(a), we highlight the cells that violate OW (green color), DDA (light green color), IW1 (yellow color) and IW2 (brown color) rules. Figure 3.8(b) shows the displacement

of corresponding cells in post-detailed placement legalization. We observe that our DFPlacer fixes the DDA violation by flipping one of the violating cells; the IW1, IW2 and OW violations are all resolved by moving the violating cells or their neighbor cells within and/or across rows.

3.1.4 Conclusion

In this work, we have proposed a scalable detailed placement legalization flow for complex FEOL constraints arising at the N10 foundry node. These include drain-drain abutment, minimum implant width, and minimum OD jogging rules. Given initial (timing-driven) placements, our $DFPlacer$ fixes 99% of DRVs with 3% increase in wirelength and minimal impact on timing. We feel that our use case of fixing all but a few tens of violations, with a highly parallelizable two-iteration strategy, is a good practical tradeoff between runtime complexity and DRV fixing. Further, the level of DRV fixing achieved by DFPlacer is encouraging, given that our default experimental configuration makes no attempt at "correctness by construction". Using OpenMP, we confirm that our flow is scalable via a distributed optimization strategy. Additionally, we study an area-DRV tradeoff between two types of standard-cell library strategies, namely, with and without dummy poly gates.

Our future work includes (i) timing and wirelength-driven placement legalization, which we believe can be enabled by more compact optimization formulations along with a more restricted perturbation range for each cell; (ii) a "smart ECO" method for the few DRVs that remain after global placement legalization; and (iii) further investigation of the scalability of our partitioning-based distributed optimization approach. Finally, we believe that our present placement-centered work may converge with such recent routing-centered works as [60], leading eventually to an "optimal detailed P&R" that can shield physical design teams from impacts of increasing ground rule complexity at N10 and beyond.

3.2 Vertical M1 Routing-Aware Detailed Placement for Congestion and Wirelength Reduction in Sub-10nm Nodes

In tandem with aggressive pitch scaling in sub-10nm technology nodes, the detailed routing problem has become extremely challenging. Routing today must deal with large numbers of complex

design rules that are driven by patterning technologies – notably, self-aligned multiple patterning and line-end cut on minimum-pitch metal layers, as well as contact- and via-layer patterning. The quest to scale "PPAC" (power, performance, area, cost) has led to a very delicate balancing act among power delivery, routing resource, and resistivity in middle-of-line (MOL) and local metal layers.

To address these challenges, the industry has seen rapid innovation in standard-cell architecture starting at the foundry $10nm$ ("N10") node, and accelerating into the N7/N5 enablement. As examples of cell architecture evolution, metal layers below M1 are used for internal routing within a standard cell, or horizontal M1 power/ground pins are removed to gain additional routing resources for inter-cell routing. These new cell architectures, wherein *inter-row M1 routing is allowed*, force new consideration of vertical alignment of cells.

New Cell Architectures in Sub-10nm

Figure 3.9 illustrates inverter (INV) layout in three types of cell architectures: (a) conventional 12-track, (b) *ClosedM1* 7.5-track, and (c) *OpenM1* 7.5-track. The conventional 12-track INV has power/ground (VDD/VSS) in M1, which prevents use of vertical M1 routing for pin access. In other words, with the conventional cell architecture, pin access is available only with M2 routing. However, in sub-$10nm$ nodes, where metal layers below M1 are used for internal cell routing, the M1 layer can be used for pin access as well as for routing with both the *ClosedM1* and *OpenM1* cell architectures.

***ClosedM1* standard cell architecture.** A *ClosedM1* standard cell has 1D vertical M1 pins, including VDD/VSS pins, as shown in Figure 3.9(b). The M1 VDD/VSS pins at the left and right boundaries of the cell are connected to M2 VDD/VSS pins at the top and bottom boundaries by using via V12. In this way, VDD/VSS pins do not block inter-row M1 routing. Also, due to the design rules for self-aligned multiple patterning (SAMP), the M1 pins in *ClosedM1* have 1D shapes and are regularly placed with a fixed pitch. In particular, the *ClosedM1* cell library that we use in this work has M1 pitch equal to the width of a placement site. Therefore, if we vertically align pins of given net, these pins can be connected by a small M1 segment with negligible routing cost or overheads. Figure 3.10(a) illustrates an example of direct vertical M1 routing (dM1) between two INVs. Here we define a direct vertical M1 routing as a (sub)net routing using only one M1 routing segment. Importantly, even though the

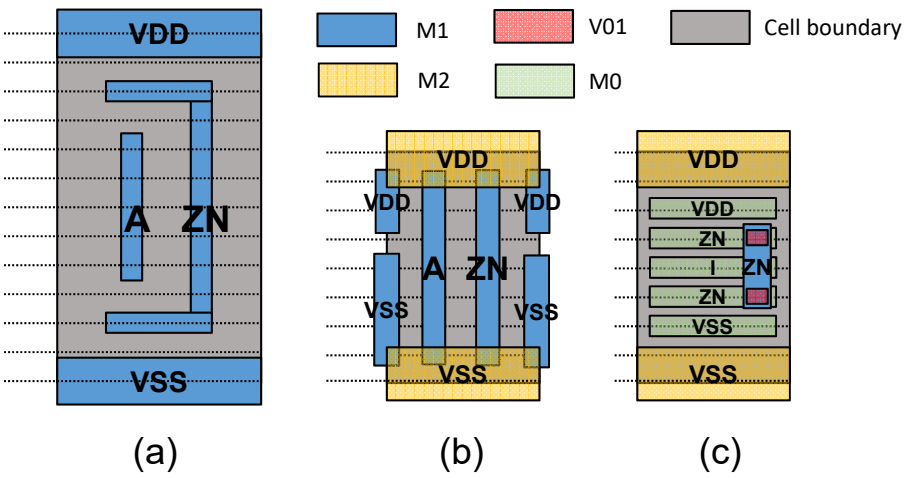


Figure 3.9: New cell architectures to gain additional routing resources. (a) Conventional 12-track INV; (b) *ClosedM1* 7.5-track INV; (c) *OpenM1* 7.5-track INV.

ClosedM1 cell architecture enables inter-row M1 routing, the realized power/performance/area (PPA) benefit from M1 routing may not be significant unless a router can effectively exploit the availability of direct vertical M1 routing. This is because M1 routing tracks are blocked by M1 pins, and the inter-row M1 routing can be used only when two pins are sufficiently aligned. Thus, *both* the detailed placer and the router must comprehend vertical alignment in order to maximally exploit direct vertical M1 routing for *ClosedM1*-based designs.

***OpenM1* standard cell architecture.** At sub-10nm nodes, the *OpenM1* standard cell architecture is introduced to enable more M1 routing resource than with the *ClosedM1* architecture. For *OpenM1* cells, M1 routing is "open" since most of the pins are on the M0 layer, which is a complementary layer below the M1 layer. As shown in Figure 3.9(c), the I, ZN, VDD, VSS pins have horizontal M0 segments, and an M1 segment connects two M0 segments for the ZN pin. We note that there is no connection between M0 and M2 segments for VDD/VSS pins. Thus, M1 routing for VDD/VSS pins must be accomplished with a special structure for the power distribution network.[22] In terms of signal routing, if two pins are overlapped horizontally (i.e., their projections onto the x-axis intersect), direct vertical M1 routing can be used to connect them. Figure 3.10(b) shows a direct vertical M1 routing between the ZN pin of the upper INV and the I pin of the lower INV. As long as the ZN and I pins are overlapped horizontally, the two pins

[22]For example, vertical M1 segments must be inserted with a fixed pitch to staple M2 and M0 VDD/VSS pins.

can be connected using a single vertical M1 segment along with two V01 vias.

Compared to both the conventional and the *ClosedM1* cell architectures, *OpenM1* effectively enables an additional metal layer for routing, which can have considerable routability benefits. Furthermore, unlike with the sub-10nm *ClosedM1* architecture, conventional P&R tools can easily find benefits from *OpenM1* without any special optimization to maximize M1 routing. This being said, below we explore the question of whether there might still be room (beyond the current state of the art in commercial P&R tooling) to optimize for better pin accessibility in *OpenM1*-based designs, given that pins are horizontal. For instance, by maximizing "overlap" between pins in a net, we might induce a router to use more direct vertical M1 routing between pins, which would reduce usage (blockage) and detouring on upper layers (M2, M3, etc.). In Section 3.2.4, we report experimental results with and without a detailed placement optimization that maximizes pin overlaps for *OpenM1*.

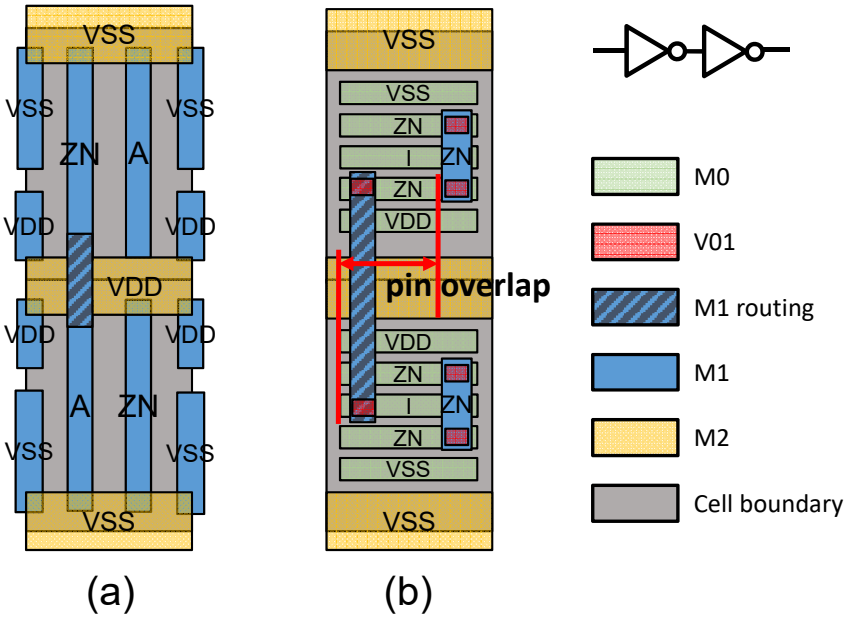


Figure 3.10: Direct vertical M1 routing examples: (a) *ClosedM1* and (b) *OpenM1*.

This Work

In this work, we propose a vertical M1 routing-aware detailed placement optimization based on mixed-integer linear programing (MILP) for two new sub-10nm cell architectures, i.e., *OpenM1* and

ClosedM1. We note that the vertical M1 routing-aware detailed placement is a completely different problem from traditional wirelength-driven detailed placement, in the sense that the routing cost is non-monotonic due to vertical M1 routing, which is almost "free". Our MILP formulation enables exploration of the tradeoff between minimization of the traditional half-perimeter wirelength (HPWL) objective and maximization of the number of vertical pin alignments (= potential direct pin-pin routings using vertical M1) via a weighting factor (α). Below, we specifically study the impact of α on routed wirelength. The main contributions of our work are summarized as follows.[23]

- We propose an MILP-based detailed placement optimization for two cell architectures that are relevant in sub-$10nm$ process nodes, to consider and exploit (direct vertical) inter-row M1 routing.
- We propose a distributable window-based optimization to overcome the runtime limitation of the MILP-based approach.
- We implement our proposed approach in C++ with OpenAccess 2.2.43 [193] and incorporate it into a commercial tool-based placement and routing (P&R) flow. The results from our approach are evaluated using a commercial tool flow.
- We explore various metaheuristic configurations (optimization degrees of freedom, window size, iteration strategy, etc.) and study impacts on runtime and solution quality.

3.2.1 Related Work

We classify relevant previous works on detailed placement and placement legalization into three categories: (i) dynamic programming-based approaches, (ii) graph model-based approaches, and (iii) MILP-based approaches. Our present work is most closely related to the third category.

Dynamic programming-based approaches. Dynamic programming (DP) has been a popular framework, particularly for row-based detailed placement, for many years. Kahng et al. [92] propose a HPWL-driven ordered single-row detailed placement with free sites. Gupta et al. [56] propose a DP-based single-row placement optimization to enable sub-resolution assist feature insertion for improved

[23]The MILP formulation will differ according to the standard cell template and layer directionality. However, our distributable optimization and exploration of metaheuristic configurations can apply with any technology.

manufacturability. Subsequent work addresses a 2D formulation [57], using DP in which vertical and horizontal costs are calculated with restricted perturbations. Hur and Lillis [70] propose a DP-based *optimal interleaving* for intra-row optimization in detailed placement. For double-patterning-aware detailed placement, Gupta et al. [52] propose a DP-based algorithm that solves coloring conflicts while minimizing the displacement of timing-critical cells.

Graph-based approaches. A literature of graph model-based approaches typically formulates placement optimization as a shortest-path computation in an appropriate directed graph. Kahng et al. [89] legalize placement of a single row with various minimization objectives, by calculating a shortest path in a directed acyclic graph constructed from the input ordering of cells. The work of [164] proposes a triple-patterning-aware detailed placement using a graph model. The authors formulate a graph to determine cell locations as well as coloring solutions for a single row placement. Du and Wong [39] address the abutment of source and drain in FinFET-based cell placement. The authors propose a graph model that captures cell flipping and adjacent-cell swapping as underlying operations for detailed placement perturbation. A shortest-path algorithm then minimizes the cost induced from the source-drain abutment. Lin et al. [106] propose a graph-based detailed placement to resolve inter-row middle-of-line conflicts. Similar to [39], a graph is constructed to handle cell flipping, swapping and shifting operation for *local reordered single row refinement*.

MILP-based approaches. While DP-based and graph model-based approaches are efficient for single-row placement, it is not easy to handle multiple-row placement optimizations (specifically, in the context of this work, vertical M1 routing-aware placement) with these approaches due to interaction between vertically adjacent cells. However, several mixed integer-linear programming (MILP)-based approaches have been proposed which handle both single-row and multiple-row placement. Lin and Chu [105] formulate a MILP for triple-patterning-aware detailed placement. The MILP is used to assign a coloring solution for each standard cell and determine the location of each cell in a single row, while minimizing placement perturbation and coloring conflicts. Li and Koh [100] propose MILP-based detailed placement approaches using *single-cell-placement (SCP) variables*. The SCP variables correspond to locations, orientations as well as placement sites of each cell. The MILP determines the best SCP variable for each cell. The same authors' extension [101] supports mixed-size circuits and improves runtime by

bounding solution spaces. Han et al. [61] adopt the MILP model of [100] [101] and extend it to support N10-relevant design rules. Further, a distributable optimization is proposed based on partitioning of the layout into windows that can be independently legalized. In our present work, we use a similar strategy as the work of [61], extending it to handle vertical M1 routing for new cell architectures in sub-10nm. Overall, our work is distinguished from previous (MILP-based) approaches in that (i) we formulate inter-row cell alignment to maximize direct vertical M1 routing, which has not been addressed in previous works, and (ii) we improve the distributable optimization of [61] by a smart selection of target windows along with a metaheuristic strategy.

3.2.2 MILP-based Optimization

In this section, we give our problem statement, followed by MILP formulations for vertical M1 routing-aware detailed placement optimization with two sub-10nm cell architectures, *OpenM1* and *ClosedM1*.

Vertical M1 Detailed Placement

Given: a post-routed placement, and per-cell placement perturbation range.

Perform: Perturb the input placement to optimize a weighted sum of (minimized) HPWL and (maximized) inter-row pin alignments, while satisfying cell location perturbation bounds and placement legality constraints.

MILP Formulation for ClosedM1

We formulate an MILP for our detailed placement problem for the *ClosedM1* cell architecture. In the following, we use notation as described in Table 3.4. For a given input layout, our objective is to minimize the weighted sum of HPWL of all nets subtracted by the total number of pin alignments for direct vertical M1 routing, while achieving a legal placement (no overlap of cells).

Minimize: $$-\alpha \cdot \sum d_{pq} + \sum_{n \in N} \beta_n \cdot w_n \tag{3.18}$$

Subject to:

$$w_n = x_{max,n} - x_{min,n} + y_{max,n} - y_{min,n}, \ \ \forall n \in N \tag{3.19}$$

$$x_{max,n} \geq x_c + x_p, \ \ x_{min,n} \leq x_c + x_p$$

$$y_{max,n} \geq y_c + y_p, \ \ y_{min,n} \leq y_c + x_p$$

$$\forall p \in P_n, \text{ where } c \text{ is the owner cell of pin } p \tag{3.20}$$

$$(x_c + x_p) - (x_{c'} + x_q) \leq G(1 - d_{pq})$$

$$(x_c + x_p) - (x_{c'} + x_q) \geq -G(1 - d_{pq})$$

$$(y_c + y_p) - (y_{c'} + y_q) \leq G(1 - d_{pq}) + H$$

$$(y_c + y_p) - (y_{c'} + y_q) \geq -G(1 - d_{pq}) - H$$

$$\forall (p, q) \text{ in } n, \text{ where } c, c' \text{ are owners of pins } p, q \tag{3.21}$$

$$\sum_{k \in K_c} \lambda_c^k = 1, \ \ \forall c \in C \tag{3.22}$$

$$f_c = \sum_{k \in K_c} f_c^k \lambda_c^k, \ \ \forall c \in C \tag{3.23}$$

$$x_c = \sum_{k \in K_c} x_c^k \lambda_c^k, \ \ y_c = \sum_{k \in K_c} y_c^k \lambda_c^k, \ \ \forall c \in C \tag{3.24}$$

$$s_{crq} = \sum_{k \in K_c} s_{crq}^k \lambda_c^k, \ \ \forall c \in C \tag{3.25}$$

$$\sum_{c \in C} s_{crq} \leq 1, \ \ \forall q \in Q, r \in R \tag{3.26}$$

HPWL calculation. Constraint (3.19) calculates the HPWL for each net n, where HPWL as usual corresponds to the half-perimeter of the minimum bounding box that contains all pins of n. The maximum and minimum x, y coordinates of pins of the net n are obtained by Constraint (3.20). The absolute coordinates of pin p is determined by adding the coordinates (x_c, y_c) of p's owner cell c to (x_p, y_p).

Table 3.4: Notations.

Notation	Meaning
d_{pq}	a binary indicator of whether pins p and q are aligned (*ClosedM1*) or overlapped (*OpenM1*)
w_n	half-perimeter wirelength (HPWL) of net n
α	a weighting factor for direct vertical M1 routing
β_n	a weighting factor for HPWL of net n
C, R, Q	sets of cells, rows, columns (placement sites)
N	set of nets
$x(y)_{min,n}$ $x(y)_{max,n}$	minimum x (y) and maximum x (y) coordinates of net n
P_n	set of pins in net n
G	a large positive constant number
H	placement row height
$x_c(y_c)$	x (y) coordinate of the center of cell c
$x_p(y_p)$	relative x (y) coordinate of pin p to its owner cell's x (y) coordinate
$x_{min,p}$ ($x_{max,p}$)	minimum (maximum) x coordinate of pin p relative to its owner cell's x coordinate
f_c	a binary indicator of whether cell c is flipped
s_{crq}	a binary indicator of whether cell c occupies site (r, q)
K_c	a set of candidates of cell c
λ_c^k	a binary indicator of whether candidate k for cell c is selected
$x_c^k(y_c^k)$	x (y) coordinate corresponding to λ_c^k
f_c^k	f_c corresponding to λ_c^k
s_{crq}^k	s_{crq} corresponding to λ_c^k
γ	maximum allowed length for a direct vertical M1 routing (unit: number of placement rows)
v_{pq}	a binary indicator of whether pins p and q are within a given range (γ) in y direction
o_{pq}	length of overlap between pins p and q
δ	minimum required overlap length for direct vertical M1 routing
ϵ	a weighting factor for the sum of overlap lengths (o_{pq})

Checking pin alignment. Constraint (3.21) checks whether pins p, q are aligned, by comparing their absolute coordinates. If the (absolute) x coordinate of p, q are not the same, $d_{pq} = 0$. Otherwise, the left side of the first and second constraints in Constraint (3.21) becomes zero, which makes $d_{pq} = 1$ allowed. In our implementation, we always ensure $d_{pq} = d_{qp}$.

Placement of each cell. Similar to [61], we assume that a perturbation range is given for each cell c, and that a cell cannot move beyond its given perturbation range. As in [61], we adopt the single-cell-placement (SCP) model of [101] to represent each candidate location and orientation for a cell. The binary variable λ_c^k represents a candidate k for a cell c, including the coordinates (x_c^k, y_c^k), the orientation (f_c^k), and whether placement site (r, q) is occupied (s_{crq}^k). These relations are handled by Constraints

(3.23), (3.24) and (3.25). Constraint (3.22) ensures that exactly one candidate is chosen for cell c among all $\lambda_c^k, k \in K_c$. Constraint (3.26) ensures a legal placement.

MILP Formulation for OpenM1

To maximize direct vertical M1 routing for the *OpenM1* cell architecture, we must maximize "overlap" between target pins, which is different from the objective for *ClosedM1*. In addition to maximizing the number of overlapping pin pairs, we also maximize the sum of overlap lengths of each pin-to-pin (sub)net so as to increase the probability that the router completes the direct vertical M1 routing. The *OpenM1* objective is:

$$\textbf{Minimize:} \quad -\alpha \cdot \sum d_{pq} - \epsilon \cdot \sum o_{pq} + \sum_{n \in N} \beta_n \cdot w_n \tag{3.27}$$

To support *OpenM1*, we slightly modify the previous MILP formulation for *ClosedM1* by introducing extra variables. In this case, d_{pq} becomes a binary indicator of whether pins p and q are "overlapped", and Constraint (3.21) is replaced with Constraints (3.28)–(3.30). Our notation is again as described in Table 3.4.

$$\begin{aligned} &a \geq x_c + x_{min,p}, \;\; a \geq x_{c'} + x_{min,q} \\ &b \leq x_c + x_{max,p}, \;\; b \leq x_{c'} + x_{max,q} \\ &\qquad \forall p, q, \text{ where } c, c' \text{ are the owner cells of pins } p, q \end{aligned} \tag{3.28}$$

$$\begin{aligned} &(y_c + y_p) - (y_{c'} + y_q) \leq G \cdot v_{pq} + \gamma \cdot H \\ &(y_c + y_p) - (y_{c'} + y_q) \geq -G \cdot v_{pq} - \gamma \cdot H \\ &\qquad \forall p, q, \text{ where } c, c' \text{ are the owner cells of pins } p, q \end{aligned} \tag{3.29}$$

$$a \geq x_c + x_{min,p}, \;\; a \geq x_{c'} + x_{min,q}$$

$$o_{pq} \leq b - a - \delta + G(1 - d_{pq}), \;\; o_{pq} \leq G \cdot d_{pq}$$

$$o_{pq} \geq -G(1 - d_{pq})$$

$$\forall (p, q) \text{ pin pairs in net } n, \forall n \tag{3.30}$$

$$d_{pq} + v_{pq} \leq 1, \;\; \forall p, q \tag{3.31}$$

Checking pin overlaps. Constraint (3.28) calculates the length of overlap in x direction between pins p and q. It first identifies the left side (a) and the right side (b) of the overlap between pins p and q. The overlap length o_{pq} is determined by a and b in Constraint (3.31). Constraint (3.29) checks whether the absolute difference of y coordinates of pins p and q is *larger* than γH and, if so, forces $v_{pq} = 1$. γ is a user-defined value for the maximum allowed vertical span of a direct vertical M1 routing.[24]

We use $\gamma = 3$, which means that a direct vertical M1 routing can cross three placement rows. For the case $v_{pq} = 1$, we do not need to make overlaps in the x direction since pins are multiple rows apart vertically; in such cases, it is difficult (i.e., highly improbable) to make a direct vertical M1 routing across multiple rows. Thus, Constraint (3.31) forces $d_{pq} = 0$ if $v_{pq} = 1$ so that the optimization does not make unnecessary overlaps. Constraint (3.30) forces $d_{pq} = 1$ if $b - a$ is larger than a predefined δ, which is the minimum required overlap length. Then, the o_{pq} is bounded by $b - a - \delta$. Otherwise, o_{pq} is bounded by zero.

3.2.3 Overall Flow

We now describe the overall flow of our optimization.

Distributable Optimization

In practice, the most critical limitation of the MILP-based approach is runtime. To overcome the runtime limitation, we adopt the distributable optimization proposed in [61].

We partition the layout into small windows (with width b_w, height b_h and optimize these windows

[24] For example, $\gamma = 1$ means that direct vertical M1 routing can traverse between two adjacent cell rows, and $\gamma = 2$ (resp. 3) means that direct vertical M1 routing can go through one (resp. two) intervening cell row(s).

in several iterations. In each iteration, we select a subset of windows that are independently optimizable, and optimize them in parallel. More specifically, we select windows that do not have any horizontal or vertical overlap (i.e., have disjoint projections onto the x-axis and onto the y-axis). For example, as shown in Figure 3.11, windows that are diagonally adjacent can be selected and optimized in parallel. This is because a given window's optimization is unaware of cell displacements concurrently being made outside of the window; if windows share projections onto the x- or y-axis, the impact of solutions on HPWL from each window cannot be accurately captured.

Figure 3.12 illustrates two example cases of (a) target windows with intersecting projections (on the y-axis) and (b) target windows with disjoint projections. Since the target windows are optimized in parallel, the optimizer calculates $\Delta HPWL1$ for the displacement of p in $w1$ without knowing pin q's displacement, and vice versa ($\Delta HPWL2$ for q in $w2$). However, according to the final locations of p and q, the pins that determine the bounding box corresponding to HPWL can change, as shown in the figure. In the (a) case, this results in a discrepancy between the total $\Delta HPWL$ and the sum of $\Delta HPWL$ from each window. In the (b) case, since p and q always determine the top-left point and the bottom-right point of the bounding box, the sum of $\Delta HPWL$ from each window is equal to the total $\Delta HPWL$.

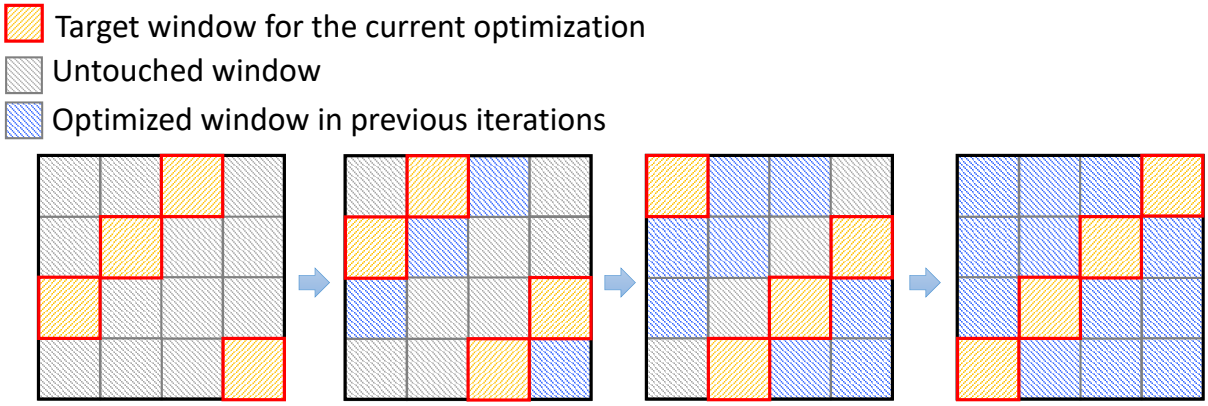


Figure 3.11: Illustration of distributable optimization.

Overall Flow

Algorithm 11 ($VM1Opt()$) gives the metaheuristic outer loop of our detailed placement optimization considering direct vertical M1 routing. The inputs include a routed layout T, a weighting factor α

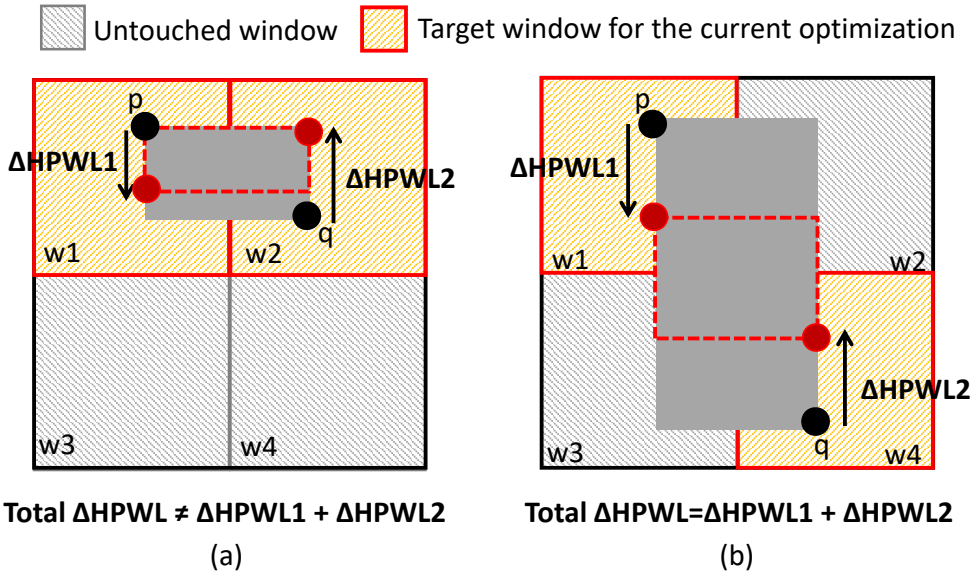


Figure 3.12: HPWL calculation for two cases. (a) Target windows with intersecting projections on the y-axis. (b) Windows with disjoint projections. In the case of (a), the total $\Delta HPWL$ is not equal to the sum of $\Delta HPWL$ values that are calculated from each window.

and a sequence(queue) of input parameter sets U. Each parameter set in U includes window width (b_w), window height (b_h), maximum x displacement for cells (l_x), and maximum y displacement of cells (l_y). The sequence U is determined empirically based on experimental results (see Section 3.2.4). The output is an optimized layout T_{opt} with a heuristically minimized objective value Obj.

In Line 2, we obtain the first input parameter set u in the current U. In Lines 3–11, we iteratively run $DistOpt()$ with u until the normalized improvement (ΔObj) of the objective with respect to Obj of the previous iteration is less than a threshold θ. We use $\theta = 1\%$ as the threshold. In Line 4, we first store the previous Obj value as $preObj$. In Lines 6–7, we then perform $DistOpt()$ with window size and perturbation range defined in u (i.e., $u.b_w$, $u.b_h$, $u.l_x$, $u.l_y$) but without allowing the flip operation ($f = 0$). After that, $DistOpt()$ is performed again in Lines 8–9, with allowing of the flip operation ($f = 1$) but without allowing perturbation. Empirically, we observe that a sequential optimization that performs perturbation and flipping serially is faster than an optimization that performs perturbation and flipping simultaneously, while both optimizations give similar solution quality. In Line 10, we update the x and y shift values for windows (t_x, t_y). Although we avoid interference between windows by selecting diagonally-adjacent windows (recall Figure 3.11) for parallel optimization, cells at the boundary (i.e., cells

that overlap two windows simultaneously) cannot be optimized. Thus, similar to the method of [61], we shift the windows to handle the unoptimized boundary region of the previous iteration. If ΔObj is less than θ (Line 3), we change u to the next input parameter set in U (Line 2). We iterate the optimization until we reach the last input parameter set in U.

Algorithm 12 describes details of $DistOpt()$. According to the given input parameters, we partition the layout into small windows (Line 1). We then select target windows that are independently optimizable and store them in D (Line 3) as explained above. Since we select target windows such that windows do not have any vertical or horizontal overlaps, the parallel optimization has $k = \sqrt{|W|}$ iterations, where $|W|$ is the total number of windows. In Lines 5–6, all windows $d \in D$ are optimized in parallel. For each window, we list candidates for each cell according to a given perturbation range (i.e., l_x and l_y, the maximum displacement of x and y, respectively). Along with input parameters α, β_n, γ and δ, we formulate the MILP instance for the window and use *CPLEX* to solve the MILP instance. The solution is updated for each window, and is then used as a boundary condition for the target windows in the next iteration.

Algorithm 11 Overall flow of *VM1Opt*

Procedure *VM1Opt*(T, α, U)
Input : Layout T, weighting factor α, queue of parameter sets U
Output : Layout T_{opt}

1: **while** $U \neq \emptyset$ **do**
2: $u \leftarrow U.pop()$; $\Delta Obj \leftarrow \infty$;
3: **while** $\Delta Obj \geq \theta$ **do**
4: $preObj \leftarrow Obj$;
5: $l_x \leftarrow u.l_x$; $l_y \leftarrow u.l_y$; $f \leftarrow 0$;
6: $(T, Obj) \leftarrow DistOpt(T, t_x, t_y, u.b_w, u.b_h, l_x, l_y, f, \alpha)$;
7: $l_x \leftarrow 0$; $l_y \leftarrow 0$; $f \leftarrow 1$;
8: $(T, Obj) \leftarrow DistOpt(T, t_x, t_y, u.b_w, u.b_h, l_x, l_y, f, \alpha)$;
9: Update t_x, t_y
10: $\Delta Obj \leftarrow (preObj - Obj)/preObj$;
11: **end while**
12: **end while**
13: $T_{opt} \leftarrow T$;
14: return T_{opt};

Algorithm 12 Procedure *DistOpt*

Procedure $DistOpt(T, t_x, t_y, b_w, b_h, l_x, l_y, f, \alpha)$
Input : Horizontal (vertical) offset t_x (t_y), width (height) of window b_w (b_h), perturbation range in x (y) l_x (l_y), binary indicator of whether flip operation is allowed f, weighting factor α
Output : Updated layout T_{opt}, objective value Obj

A set of windows $W \leftarrow Partition(T, t_x, t_y, b_w, b_h)$;
for $i = 1$ to $\sqrt{|W|}$ **do**
  $D \leftarrow$ set of current target windows;
  // parallel optimization
  $MILPFormulation(d, l_x, l_y, f, \alpha)$ for $\forall d \in D$;
  Solve MILP and update MILP solutions to T;
  // parallel optimization ends
end for
$T_{opt} \leftarrow T$
$Obj \leftarrow CalculateObj(T_{opt})$;
return T_{opt};

3.2.4 Experimental Setup and Results

Experimental Setup

We implement our flow in C++ with *OpenAccess 2.2.43* [193] to support LEF/DEF [186], and with *IBM ILOG CPLEX Optimization Studio v12.6.3* [180] as our MILP solver. We apply our detailed placement optimization flow to ARM Cortex M0 core (*M0*) and three designs (AES, JPEG and VGA) from the *OpenCores* website [191]. The design information is summarized in Table 3.5. The four designs are implemented with $7nm$ *OpenM1* and *ClosedM1* triple-V_t libraries from a leading technology consortium. We synthesize the testcases using *Synopsys Design Compiler K-2015.06-SP4* [195], and then perform placement and routing using *Cadence Innovus v16.1* [174]. The experiments are performed with 8 threads on a $2.6GHz$ Intel Xeon dual-CPU server. We note that with flexible computing resources, the number of usable threads could be as large as the number of layout windows that are independently optimizable ($\sqrt{|W|}$) to reduce runtime for larger designs.

Experimental Results

We have conducted two basic types of experiments. **Expt1** experiments seek to optimize our overall flow by finding input parameters and optimization sequences that give dominating runtime versus solution

quality tradeoffs. The AES design with *ClosedM1* is used for **Expt1** experiments. **Expt2** experiments apply our flow to both *ClosedM1*-based and *OpenM1*-based designs. For all experiments, we use $\beta = 1$ so that our MILP formulation minimizes pure HPWL.

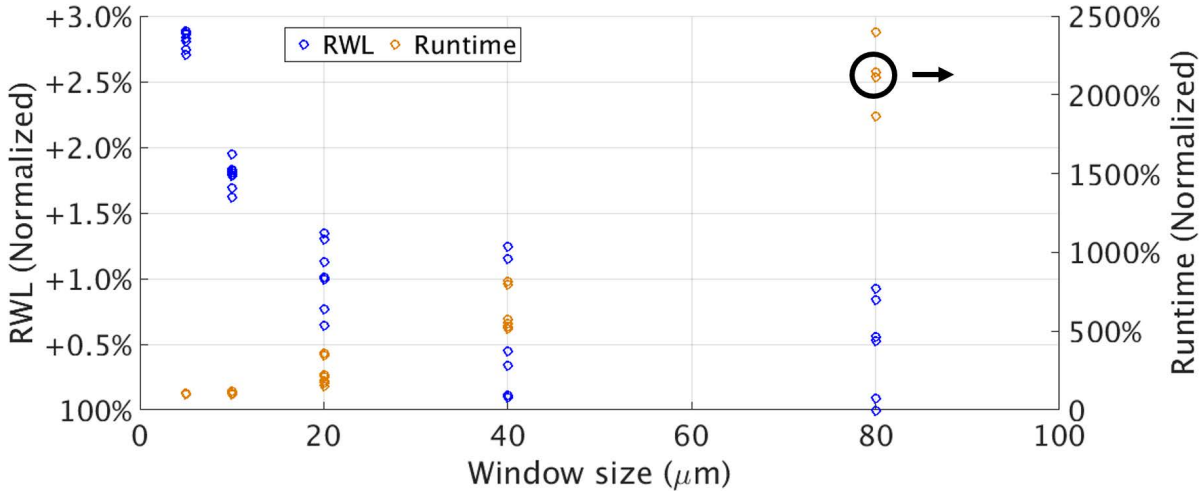


Figure 3.13: Scalability test with various window sizes and perturbation ranges.

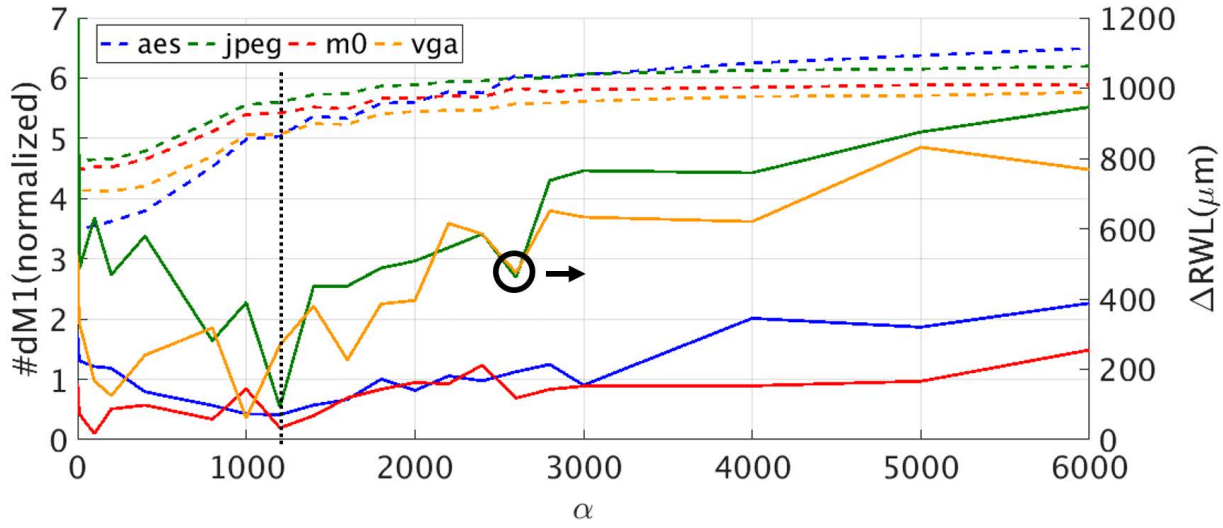


Figure 3.14: Sensitivity of total routed wirelength (RWL) and the number of direct vertical M1 routings (#dM1) to α.

Expt1-1: Scalability study on window size and perturbation range. We sweep the window size and the perturbation range to study the tradeoff between solution quality and runtime. We assume square windows and vary $b_w = b_h$ from $5\mu m$ to $80\mu m$. For the perturbation range, we try $l_x \in \{2, 3, 4, 5\}$,

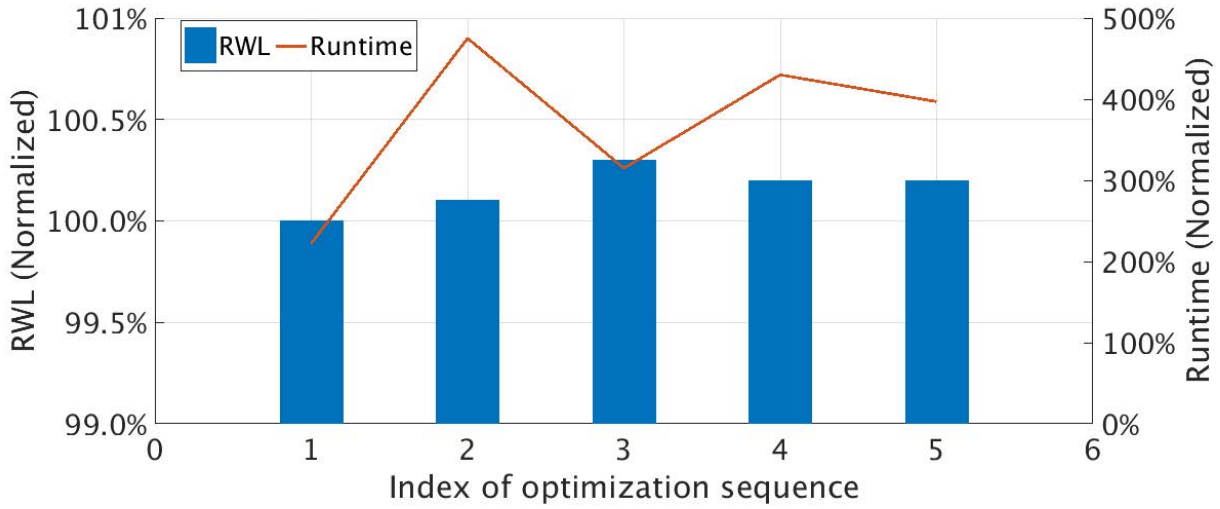


Figure 3.15: Results of various optimization sequences.

$l_y \in \{0, 1\}$. In this experiment, we only run one iteration in Algorithm 11 (i.e., one pair of $DistOpt()$). Figure 3.13 shows the normalized routed wirelength (RWL) and runtime versus the window size. As the window size increases, the routed wirelength decreases, as expected. However, we observe huge runtime increases, e.g., 5× runtime increase with $b_w = b_h = 40\mu m$. To compromise the runtime overhead and the solution quality, we select the option with shortest runtime that gives $\leq$ 1% total routed wirelength increase compared to the minimum routed wirelength; this is $b_w = b_h = 20\mu m$, $l_x = 4$, and $l_y = 1$.

Expt1-2: Sensitivity study for α. We sweep α values and study the impact of α on the number of direct vertical M1 routings (#dM1) and the routed wirelength (RWL). We vary α from 0 to 6000 – e.g., for *ClosedM1*-based design, the objective with $\alpha = 10$ prefers one more aligned pin pair at the cost of at most 10 units increase in HPWL. Figure 3.14 shows total routed wirelength (RWL) and the number of direct vertical M1 routings (#dM1) versus α. As α increases, the number of direct vertical M1 routings increases. However, maximizing the number of direct vertical M1 routings does not always reduce routed wirelength, Based on our studies, we select $\alpha = 1200$ for *ClosedM1*. Similarly, we experiment on *OpenM1*-based designs and select $\alpha = 1000$.

Expt1-3: Sequence of optimization. We explore various sequences of input parameter sets ($b_w = b_h$, l_x, l_y) to optimize our overall flow. We illustrate this with five example optimization sequences: (1) (20, 4, 1); (2) (10, 3, 1) → (10, 4, 0) → (20, 4, 0) ; (3) (10, 3, 1) → (20, 3, 1) → (20, 3, 0); (4) (10, 3,

1) → (20, 3, 0) ; and (5) (10, 3, 1) → (10, 3, 0) → (20, 3, 1) → (20, 3, 0). Figure 3.15 shows RWL and runtime for these optimization sequences. We observe that optimization sequences 1 and 2 with $l_x = 4$ give better solution quality (in terms of RWL). However, optimization sequence 2 consumes twice the runtime of optimization sequence 1. Therefore, $(20, 4, 1)$ would be a preferred choice of sequence.

Table 3.5: Results of Expt2.

Design	#Inst	Util (%)	α	#dM1		M1 WL (μm)		#via12		HPWL (μm)		RWL (μm)		WNS (ns)		Power (mW)		Runtime
				Init	Final (Δ%)	Init	Final (Δ%)	Init	Final (Δ%)	Init	Final (Δ%)	Init	Final (Δ%)	Init	Final	Init	Final (Δ%)	(sec)
***ClosedM1*-based designs**																		
M0	9922	75%	1200	545	2955 (442.2)	676	629 (-7.0)	35766	31932 (-10.7)	22850	23760 (4.0)	27636	26833 (-2.9)	0.000	0.000	2.444	2.431 (-0.5)	344
AES	12345	75%	1200	631	3177 (403.5)	970	710 (-26.8)	43248	38631 (-14.4)	30420	28890 (-5.0)	32560	30471 (-6.4)	0.000	0.000	3.240	3.212 (-0.9)	711
JPEG	54570	75%	1200	3694	20688 (460.0)	3605	3329 (-7.7)	179315	153500 (- 5.7)	91030	88900 (-2.3)	96621	90593 (-6.2)	0.000	0.000	28.592	28.399 (-0.7)	1216
VGA	68606	75%	1200	2460	12473 (407.0)	5973	5428 (-9.1)	270930	255466 (-10.7)	169200	169800 (0.4)	206558	204269 (-1.1)	0.000	-0.002	53.614	53.542 (-0.1)	561
***OpenM1*-based designs**																		
M0	9891	75%	1000	1183	1931 (63.2)	3681	3790 (3.0)	35099	34336 (-1.7)	24790	24570 (-0.9)	29884	29575 (-1.0)	-0.003	0.000	2.475	2.468 (-0.3)	298
AES	12348	75%	1000	1341	1975 (47.3)	4646	4620 (-0.5)	43004	42269 (-4.1)	30670	29980 (-2.2)	34338	33592 (-2.2)	0.000	0.000	3.273	3.263 (-0.3)	325
JPEG	54689	75%	1000	8391	13763 (64.0)	18709	19244 (2.8)	173622	166411 (-3.8)	92100	91110 (-1.1)	103257	101463 (-1.7)	0.000	-0.001	29.024	28.957 (-0.2)	1026
VGA	68729	75%	1000	7714	13132 (70.2)	26912	26823 (-0.3)	261424	251558 (-2.2)	170000	168700 (-0.8)	215218	213598 (-0.8)	0.000	-0.002	53.805	53.730 (-0.1)	515

Expt2-1: Detailed placement optimization for *ClosedM1*-based designs. Table 3.5 shows overall results for our detailed placement optimization. Our optimizer increases the number of direct vertical M1 routings by more than 4× compared to the initial post-routing solution, while decreasing overall M1 wirelength. This means that we remove long vertical M1 routings that are not used for direct vertical routing, while generating many short, direct vertical M1 routes; this results in smaller M1 wirelength and a larger number of M1 routing segments. Along with the increase in the number of direct vertical M1 routings, we achieve up to 6.4% routed wirelength (RWL) reduction and up to 14.4% #via12 reduction without design rule violations (DRVs).[25] Total power also decreases by up to 0.9%. For half of the designs, HPWL increases in favor of more dM1 to further reduce routed wirelength.

To study the impact of direct M1 routing on congestion reduction, we increase the initial utilization on the AES design so as to induce congestion hotspots, which lead to design rule violations. In Figure 3.16, we show that our optimizer has the added benefit of avoiding a substantial fraction of DRVs (#DRVs orig versus opt in the figure). We note that even though our optimization consistently decreases DRVs, routing QoR is ultimately determined by the initial placement quality. Notably, placement QoR with utilization 83% from the commercial tool is worse than placement with utilization 84% in terms of DRVs. The cause of this phenomenon is beyond our present scope.

[25]Here we refer to routing DRVs. In this work, we do not consider advanced node placement rules (e.g., drain-drain abutment, minimum implant area, etc.). However, our framework is fully compatible, and can be easily integrated, with the work of [61] and complex sub-14nm rules.

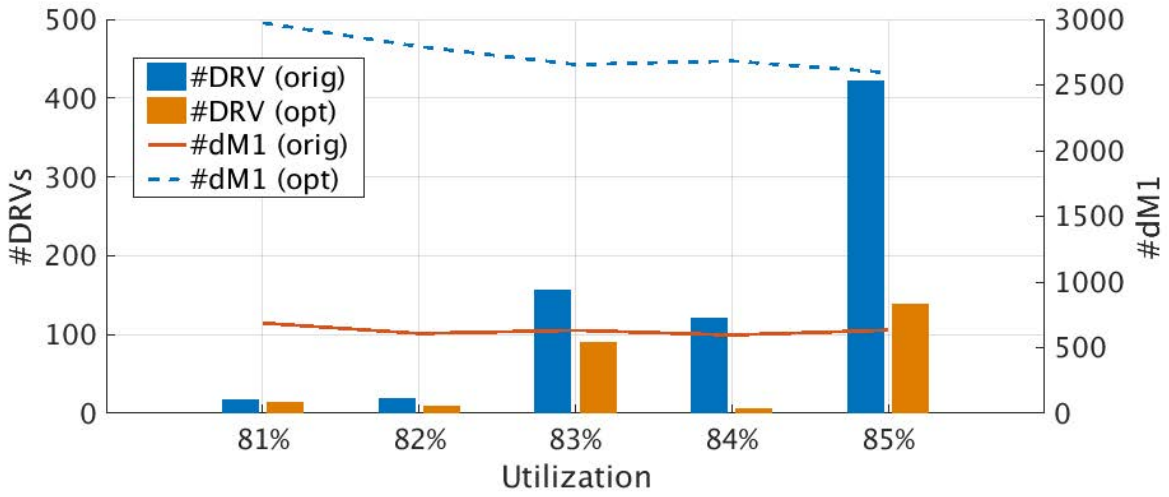


Figure 3.16: The number of DRVs after optimization for AES design with various utilizations. Also shown: the number of direct vertical M1 routings.

Expt2-2: Detailed placement optimization for *OpenM1*-based designs. Our optimizer increases the number of direct vertical M1 routings by around 60% compared to the initial post-routing solution. We observe that the increase of the number of direct vertical M1 routings for *OpenM1*-based designs is much smaller than that for *ClosedM1*-based designs. This small increase of the number of direct vertical M1 routings results in only up to 2.2% routed wirelength reduction, and up to 4.1% #via12 reduction, without design rule violations. There can be several reasons for the lesser improvement seen for *OpenM1*-based designs. Our current hypothesis is that P&R for *OpenM1* is very similar to traditional P&R in terms of pin access. In traditional P&R flows with conventional libraries, where most pins are on M1, the M2 layer is used to access the pins. Similarly, *OpenM1* cells also have pins (on or) below M1, and M1 can be used for pin access. Thus, P&R for *OpenM1* can be seen as a variant of the conventional P&R flow, where the bottom routing layer is shifted down to M1. Indeed, in *OpenM1*-based designs, direct vertical M1 routing can block access to other pins, which limits the wirelength reduction. On the other hand, in *ClosedM1*-based designs, direct vertical M1 routing does not block any pin access, and is thus "free" in terms of routing resource. Compared to *ClosedM1*, where routed wirelength can be reduced even at the cost of HPWL increase, *OpenM1*-based designs prefer smaller α to reduce HPWL. However, given our use of a black-box commercial router, it is difficult to identify root causes of the improvement difference between *OpenM1* and *ClosedM1*. This is the subject of one of our ongoing studies.

3.2.5 Conclusion

In this work, we propose a vertical M1 routing-aware detailed placement optimization based on mixed-integer linear programing (MILP) for two new cell architectures in sub-10nm nodes, i.e., *ClosedM1* and *OpenM1*. With our optimization, up to 6.4% (resp. 2.2%) total routed wirelength reductions and 14.4% (resp. 4.1%) #via12 reductions are achieved for *ClosedM1*-based (resp. *OpenM1*-based) designs, with no adverse timing impact.

We note that the library characterization model for *ClosedM1* library cells might need to change since the vertical M1 routings might affect cells' library model (change in gate capacitance, etc.). However, according to our study with an INV cell in *ASAP ASU 7nm PDK* [169], the timing impact is negligible ($\leq$ 0.1ps).[26]

Our future works include (i) a comprehensive study of timing library characterization for *ClosedM1*, to accurately capture the timing impact of direct vertical M1 routing; (ii) extension of our placement objective function to consider other design criteria, including timing criticality, pin density, routing congestion, and routing design rules; (iii) (meta)heuristic innovation to improve QoR and scalability; and (iv) theoretical understanding of *OpenM1*-based layout design to inform an improved optimization strategy.

3.3 Acknowledgments

Chapter 3 contains reprints of Peter Debacker, Kwangsoo Han, Andrew B. Kahng, Hyein Lee, Praveen Raghavan and Lutong Wang, "Vertical M1 Routing-Aware Detailed Placement for Congestion and Wirelength Reduction in Sub-10nm Nodes", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2017; and Kwangsoo Han, Andrew B. Kahng and Hyein Lee, "Scalable Detailed Placement Legalization for Complex Sub-14nm Constraints", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2015. The dissertation author is a main contributor to, and a primary author of, each of these papers.

[26] We modify pin shapes (increase the pin length by 32nm) in a cell layout, run parasitic extraction with *Calibre xRC v2016.1_31.21* [188], and measure cell delay and slew with *HSPICE I-2013.12* [196]. We observe that the delay and slew impacts of the pin modifications are negligible ($\leq 0.1ps$). Further, there are only a small number of possible uses of vertical M1 incident to a cell (this number is a function of the number of pins, and of upward versus downward alignments). In a regime where these delay and slew changes must be modeled, each of these contexts could be characterized.

I would like to thank my coauthors Peter Debacker, Kwangsoo Han, Andrew B. Kahng, Praveen Raghavan and Lutong Wang.

Chapter 4

Evaluations of Design Enablements for Advanced VLSI Technologies

In advanced technology nodes, BEOL interconnect geometry has become a key lever for design enablement and design-technology co-optimization, due to its significant impact on physical design QoR. Complex design rules along with more pervasive use of multi-patterning have increased the difficulty of maintaining high layout densities. Intuitively, emerging constraints such as unidirectional patterning or increased via spacing will decrease achievable density of the final place-and-route solution, worsening die area and product cost. Also, the rapid increase of interconnect RC leads to not only performance loss from interconnect delay increase, but circuit power and area degradation as well. Optimization of BEOL dimensions (i.e., wire width, spacing and thickness subject to a given layers pitch constraint) is crucial to achieve better product performance, power and area.

Understanding the interaction of design flows and technology choices is a crucial need for early development of BEOL process technologies. However, it is nontrivial to evaluate BEOL stack options since the routing outcomes highly depend on the input design (e.g., netlist, placement, etc.). As far as we know, no systematic methodology exists for accurate assessment of BEOL stack choice impact on physical chip implementation.

This chapter presents three distinct evaluation methodologies for design flows and technology

enablements. First, we study impacts of patterning technology choices and associated design rules on physical implementation density, with respect to *cost-optimal* design rule-correct detailed routing. A key contribution is an Integer Linear Programming (ILP) based optimal router (*OptRouter*) which considers complex design rules that arise in sub-20nm process technologies. Using OptRouter, we assess wirelength and via count impacts of various design rules (implicitly, patterning technology choices) by analyzing optimal routing solutions of clips (i.e., switchbox instances) extracted from post-detailed route layouts in an advanced technology. Second, we study BEOL interconnect stack geometry by exploring wire aspect ratio (AR) and wire line-space duty cycle (DC). We perform SPICE-based analyses of timing path delays to find delay- or power-optimal (AR,DC) combinations, and also perform block-level studies with placed and routed designs. Based on our experimental results, we provide various insights on BEOL stack geometry: (i) optimal (AR,DC) for a given wire pitch with respect to power and delay; (ii) sensitivities of optimal (AR,DC) to circuit parameters (e.g., driver strength, input slew, output load, wirelength); (iii) optimal (AR,DC) when multiple interconnect layers are considered; and (iv) potential impacts of BEOL stack optimizations within future design-aware manufacturing and/or manufacturing-aware design methodologies. Third, we propose a systematic framework to measure routing capacity of a BEOL stack as well as inherent capability of routers. Based on our experimental results, we observe consistent results across mesh-like placement and placements from various placers. Our proposed framework also enables other insights into BEOL stack options. Using our framework, we empirically study the relation between the routing hotspot size and routing failure. We also present an analytical study, based on exponentiation of a Markov transition matrix, of the impact of design size on routing failure.

4.1 Evaluation of BEOL Design Rule Impacts Using an Optimal ILP-based Detailed Router

To scale semiconductor process nodes below the resolution limits of $193i$ optical lithography, multi-patterning techniques (e.g., litho-etch-litho-etch (LELE) and self-aligned double and quadruple patterning (SADP, SAQP) [103] have already been widely used in production. Multi-patterning is expected to be the basis of mainstream process offerings through the foundry $10nm$ and even $7nm$ nodes, and will

persist even with deployment of extreme ultraviolet (EUV) lithography [185]. Although multi-patterning techniques are key enablers for advanced sub-$20nm$ process technologies, they can induce highly complex design rules which challenge both IC physical design tools and the development (and enablement) of IC physical implementation methodology. Tight design rules (e.g., via placement restrictions, unidirectional routing on Mx layers, etc.) lead to design wirelength and density overheads, to the point where benefits from technology scaling reduce or even disappear altogether. Assessing the real value of a prospective future technology is also difficult in FinFET nodes, where higher drive strengths enable smaller standard-cell footprints that further challenge pin access and routability [3].

Given the above considerations, as well as the enormous cost of technology development and design enablement for a new process node, it is critical for the industry to be able to assess the impact of design rules (implicitly, patterning technology choices) on physical implementation metrics. Such assessments should be made as early as possible, to permit correct choices among various technology options and to enable design-technology co-optimization. Unfortunately, there are two basic reasons why process technology developers cannot easily evaluate impacts of complex design rules on chip implementation metrics. First, EDA vendors often require prolonged, close co-development with customers to correctly support new advanced design rules. While the latest Library Exchange Format standard (LEF5.8) [186] supports advanced design rule descriptions, even for the rapidly approaching foundry $10nm$ node there is varying (and contradictory) support across the EDA industry today [139, 127]. Thus, it is practically difficult to study new "future" design rules with current EDA tools. Second, EDA tools apply many heuristics to perform efficient large-scale layout optimizations. This clouds evaluations of how new patterning technologies or design rules impact chip implementation metrics. In other words, the "chicken-egg" relationship between current EDA algorithms that are optimized for current design enablements (design rules, cell libraries, etc.) makes it difficult to assess true impacts of future design enablements. Wherever possible, we would like to reduce the "chicken-egg" obstacles to design rule and patterning technology assessment.

In this work, we provide a framework for evaluating how prospective sub-$20nm$ design rules – as well as back-end-of-line (BEOL) stack choices – will affect chip implementation metrics such as density or wirelength. Our framework is based on *optimal* detailed routing that is correct with respect to advanced

design rules. We describe *OptRouter*, an ILP-based optimal detailed router which considers various design rules and technology options especially for the coming $10nm$/$7nm$ process nodes. OptRouter computes optimal routing solutions for small switchboxes (approximately the size of a single gcell [88], similar to the recent work of [78]), and has the ability to consider routing direction (unidirectional or bidirectional), design rules induced by advanced patterning technology (e.g., SADP), via adjacency restrictions, and pin shapes. Our studies combine realistic testcases in multiple technologies (including testcases synthesized with a prototype $7nm$ cell library from a leading commercial IP provider) with cost-optimal detailed routing. It is this combination that enables new, quantitative assessment of design rule impact on detailed routing metrics. The key contributions of our work are summarized as follows.

- We formulate as an integer linear program (ILP) a minimum-cost switchbox routing problem that arises in advanced technology nodes (corresponding to clips from standard-cell place-and-route instances). In contrast to previous approaches, our formulation captures multi-pin net routing (i.e., Steiner routing), via shapes, via adjacency restrictions, pin shapes, layer uni-/bi-directionality, and SADP constraints that occur with sub-$20nm$ patterning.

- We develop *OptRouter*, which extracts layout clips from place-and-route solutions and uses *IBM ILOG CPLEX Optimization Studio v12.5.1* [180] to solve the corresponding ILP instances. The correctness and capability of OptRouter are validated against commercial router results with foundry $28nm$ 8- and 12-track and $7nm$ 9-track libraries.

- We apply OptRouter within a novel methodology to quantify and rank impacts of complex sub-$20nm$ design rules on layout metrics (wirelength, vias, and routability). Our testbed notably includes a prototype $7nm$ PDK from a leading IP provider, as well as the aforementioned $28nm$ foundry libraries.

- Our comparisons of different design rules' impacts can potentially guide patterning technology choices and other basic design-technology co-optimization decisions.

4.1.1 Related Work

Relevant previous works are found in two areas: (1) design rule evaluation frameworks, and (2) ILP-based global and detailed routers.

Design rule evaluation. The work of [53] exemplifies efforts to connect layout ground rules with layout area, electrical variability, and parametric yield implications. Specifically, the authors of [53] study the effect of a line-end extension rule on logic standard cell and SRAM bitcell layout area, and on leakage variability and parametric yield. Ghaida and Gupta [48] propose DRE, a platform that comprehensively connects design rule alternatives to the automated synthesis of standard-cell library cells, and then to the power-performance-area envelope of standard-cell based layouts of small blocks. Subsequently, [47] extends the DRE approach to chip-level analyses. Badr et al. [11] suggest a pattern matching-based design rule evaluation method, which is then applied to checking of routing within standard cells. A fundamental distinction between these previous works and our present work is that we provide a new capability to assess design rule and patterning technology choices *with cost-optimal detailed routing*.

ILP-based routers. ILP has been widely used for optimization problems due to its simplicity along with its ability to find optimal solutions up to some limit of tractable instance complexity. A number of works adopt ILP for global routing, often starting from a multi-commodity flow perspective. The early work of Carden and Cheng [20] uses column-generating techniques within a multi-commodity flow based global router. Cho et al. [28] propose a global router based on box expansion and progressive ILP. After decomposing nets into two-pin nets, ILP is used to choose a routing between two L-shaped candidate routings for each two-pin net within a box. The approach iteratively expands the box and solves new nets within the expanded new box, using progressive ILP and maze routing. Similarly, Hu et al. [67] use ILP for global routing; they enumerate two path candidates to connect two-pin nets after initial routing, and an ILP is formulated to select the better path between the two candidates.

An important recent work is that of Jia et al. [78], which proposes a *detailed* router based on multi-commodity flow. The authors of [78] formulate an ILP for detailed routing with all nets being two-pin nets. Pin shapes and basic design rules (side-to-side, tip-to-tip, cut-to-cut) are considered. The proposed methods are demonstrated to reduce the number of Design Rule Check violations in a $45nm$

technology without wirelength or via overheads. A fundamental distinction between the work of Jia et al. [78] and our present work is that [78], while using ILP, does not guarantee optimal routing since multi-pin nets are not handled in the formulation. Further, only basic design rules are considered. In particular, the ability to compute minimum-cost *optimal* routing solutions with SADP-specific rules and via shapes is unique to our present work.

4.1.2 Optimal Routing Formulation

We now describe our ILP-based formulation of the detailed routing problem for a netlist of multi-pin nets, with consideration of via adjacency restrictions, unidirectional routing, SADP-aware line end rules, pin shapes, and via shapes. Like previous works, our development adopts the well-known paradigm of multi-commodity flow. Table 4.1 gives the notations that we use.

Table 4.1: Notations.

Notation	Meaning
N	set of multi-pin nets
n_k	k^{th} multi-pin net
s_k	source of n_k
T_k	set of sinks of n_k
$t_{k,i}$	i^{th} sink of n_k
$G(V, A)$	routing graph
V	set of vertices (of the routing graph)
v_i	a vertex with the location (x_i, y_i, z_i)
A	set of directed arcs
$a_{i,j}$	a directed arc from v_i to v_j
$e^k_{i,j}$	0-1 indicator whether $a_{i,j}$ is used in the routing of n_k
$c^k_{i,j}$	cost for $a_{i,j}$ in the routing of n_k
$f^k_{i,j}$	flow variable for $a_{i,j}$ in the routing of n_k
$p^k_{r,i}(p^k_{l,i})$	0-1 indicator whether there are the flows connected to v_i coming from right (left) side, in the routing of n_k

General Routing Problem Formulation

We use a *routing graph* $G = (V, A)$ to represent available routing resources, e.g., metal tracks on multiple layers, and inter-layer vias. Each vertex $v_i \in V$ is associated with variables that represent coordinates in the three-dimensional routing resources: horizontal metal track x_i, vertical metal track y_i

and metal layer z_i. A directed arc $a_{i,j}$, where $x_j = x_i$, $y_j = y_i$ and $z_j = z_i \pm 1$, represents a via. We solve the optimization:

$$\textbf{Minimize:} \quad \sum_{n_k \in N} \sum_{a_{i,j} \in A} c_{i,j}^k \cdot e_{i,j}^k$$

Subject to:

$$\sum_{n_k \in N} (e_{i,j}^k + e_{j,i}^k) \leq 1 \quad a_{i,j}, a_{j,i} \in A \tag{4.1}$$

$$e_{i,j}^k \geq \frac{f_{i,j}^k}{|T_k|} \quad a_{i,j} \in A, n_k \in N \tag{4.2}$$

$$e_{i,j}^k \leq f_{i,j}^k \quad a_{i,j} \in A, n_k \in N \tag{4.3}$$

$$\sum_{v_j : a_{i,j} \in A} f_{i,j}^k - \sum_{v_j : a_{j,i} \in A} f_{j,i}^k = \begin{cases} |T_k| & \text{if } v_i = s_k, n_k \in N \\ -1 & \text{else if } v_i \in T_k, n_k \in N \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

The objective is to minimize the weighted sum of $e_{i,j}^k$, i.e., weighted total wirelength and the number of vias. Constraint (4.1) ensures that each arc is used by only one net. Constraints (4.2) and (4.3) pertain to the binary variable $e_{i,j}^k$, which indicates whether there is a flow through $e_{i,j}$. Constraint (4.4) ensures source-sink connectivities (flow conservation). The first and second terms respectively represent the sum of the flows exiting v_i (outflows of v_i) and the sum of the flows entering v_i (inflows of v_i). For any internal node that is not a source or a sink, the sum of the node's outflows must equal to the sum of the node's inflows. For a source s_k, the sum of outflows of s_k must be $|T_k|$ (the number of sinks) since there must be $|T_k|$ flows which connect between s_k and $|T_k|$ number of sinks in n_k, and the sum of inflows of s_k must be zero. On the other hand, for a sink $v_i \in T_k$, the sum of inflows must be one since a flow coming from s_k must reach each sink, and the sum of outflows must be zero.

Figure 4.1 shows a two-net example consisting of a three-pin net (n_1) and a two-pin net (n_2), along with its solution. Net n_1 has a source v_1 and two sinks, v_3 and v_4. Net n_2 has a source v_5 and a sink, v_6. According to Constraint (4.4), for n_1, the sum of outflows of the source node $v_1 = 2$ ($|T_1|$) and the

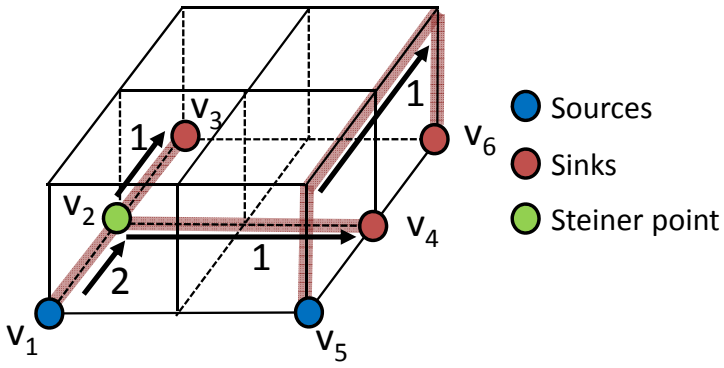


Figure 4.1: Example showing multi-pin nets and the routing solution.

sum of inflows of sink nodes v_3 and $v_4 = -1$. Similarly, for n_2, the sum of outflows of $v_5 = 1$ and the sum of inflows of $v_6 = 1$. For all other vertices, the sum of outflows is equal to the sum of inflows so that flows are conserved for each net. According to Constraint (4.1), $e^1_{1,2}$, $e^1_{2,3} = 1$, which connects between v_1 and v_3, since $f^1_{1,2}$, $f^1_{2,3}$ have non-zero values. Constraint (4.1) forces $e^2_{1,2}$, $e^2_{2,1}$, $e^1_{2,1}$ to be zero so that the edge between v_1 and v_2 can be reserved only for n_1.

Routing Rule Formulation

Via restrictions. As noted in [103], placement of vias next to each other is not allowed in advanced nodes. That is, as via pitches are larger (e.g., by a $\sqrt{2}$ factor) than metal pitches, placement of a via at a particular location blocks horizontally and vertically adjacent locations, and sometimes diagonally adjacent locations as well. We use the following constraint so that any neighbor vertical arcs $a_{i',j'}$ of a vertical arc $a_{i,j}$ can be blocked if there is a via between v_i and v_j, where $x_{i'} = x_{j'} = x_i \pm 1$, $y_{i'} = y_{j'} = y_i \pm 1$, $z_{i'} = z_i$ and $z_{j'} = z_j$.

$$e^k_{i,j} + e^k_{j,i} + e^k_{i',j'} + e^k_{j',i'} \leq 1 \qquad \forall\ a_{i',j'}$$

In our study below, we consider two types of restrictions: (i) blocking of orthogonally adjacent locations (N, E, S, W neighbors) and (ii) blocking of both orthogonally and diagonally (NE, NW, SE, SW) adjacent locations.

Unidirectional routing. Patterning with severe restriction, as with one-pitch/one-orientation metal layers, is used because of better robustness, scalability and manufacturability – as well as fewer masking steps – compared to a standard LELE-patterned bidirectional metal layer. We trivially restrict routing on a given layer to be unidirectional by removing arcs that are not in the preferred direction. (See also our discussion of SADP constraints, below.)

Pin shape. In the above example of Figure 4.1, we assume that each source or sink has a single fixed location. However, in actual routing, a pin has multiple access points, which means that the source or sink locations ultimately used in the routing solution can vary. Multiple access points for source and sink are captured by creating a *supersource* or *supersink* which is connected to all available access points in the corresponding pin. We note that the supersource and supersink are virtual vertices, which are not actually located but which nonetheless have flows. We also observe that each access point for source or sink becomes an internal node.[27]

Via shape. To trade off between manufacturability and routability, various via types with square or rectangular shapes may be instantiated. Some vias shapes, e.g., 2×2 size, are too large to be modeled as a single vertex in our routing graph. We model a via's shape by creating a *representative vertex* which is connected to all the vertices that belong to a via, according to that via's footprints on lower and upper layers.[28]

Figure 4.2(a) shows a via and its vertices in a routing graph. v_v is a square via with size 2×2 (with respect to the number of metal tracks). With the flow conservation constraint (Constraint (4.3)), once a flow (routing) enters v_v, the flow goes through one of four vertices in the upper layer. Note that for each via type, vertices are created for all possible locations where the via can be placed. For example, if a 2×2 size square via type is added to the routing graph with three layers and 15×15 tracks ($15 \times 15 \times 3$), we will create 392 ($14 \times 14 \times 2 = (15-1) \times (15-1) \times (3-1)$) vertices for the square via at all possible

[27]Pin shape is important in assessment of routing costs, e.g., smaller pin geometries with fewer access points in advanced FinFET nodes are a major challenge to detailed routing. In 7.5T or 7.25T library cells in FinFET nodes, power/ground rails, fin connections and other aspects of standard cell architectures must reconcile with pin shapes (access points). Strict tip-to-tip spacing (more than one contacted poly pitch (pin pitch)), diagonal via placement restriction as discussed above, and wider power rails also decrease the number of access points to a cell and potentially cause unroutability [104]. We study interactions of smaller pin shapes in $7nm$ (Figure 4.9(c)) and routing rules in Section 4.1.3.

[28]Doubled or redundant vias are also modelable with small modification of via shape formulation.

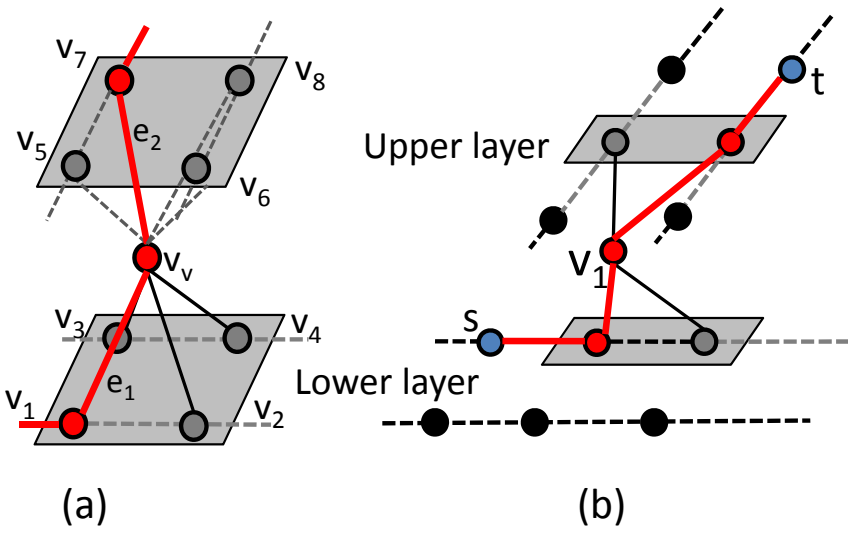


Figure 4.2: Via shape. (a) 2×2 square via. (b) 2×1 bar via.

locations. Further, note that we use lower cost values for larger via shapes so that the optimization selects as many larger vias as possible to achieve better manufacturability.

In addition to the basic formulation with Constraints (4.1)–(4.3), vertices used by a via must be blocked and not be used by other nets. For example, in Figure 4.2(a), as v_7 is selected by e_2, all the gray edges connected to other vertices used by the via (v_5, v_6, v_8) must be disabled for other nets. A generalized formulation is given in Constraint (4.5) where i' are the vertices that are not used for the routing but are within the used via shape, and j' are the neighbor vertices of i':

$$\sum_{n_k \in N} (e^k_{v,i} + e^k_{i,v}) + \sum_{n_{k'} \in N, v_{i'}, v_{j'} : a_{i',v}, a_{j',i'} \in A} (e^{k'}_{i',j'} + e^{k'}_{j',i'}) \leq 1$$

$$\text{where } a_{i,v} \in A, k' \neq k, i' \neq i \tag{4.5}$$

Thus, Constraint (4.5) prevents any other nets from using the vertices i' or the edges connected to i'. Figure 4.2(b) shows an example of 2×1 size *bar via* shape. Vertices s and t are source and sink, respectively. The red lines are selected as routing from s to t. The gray dots in Figure 4.2(b) are disabled for other nets by Constraint (4.5), so that there is no overlap between the bar via and other nets.

SADP-aware rules. Xu et al. [162] propose SADP-specific design rules. Figure 4.3(a) illustrates how the end of line (EOL) of a wire segment is the key parameter to check with such rules.

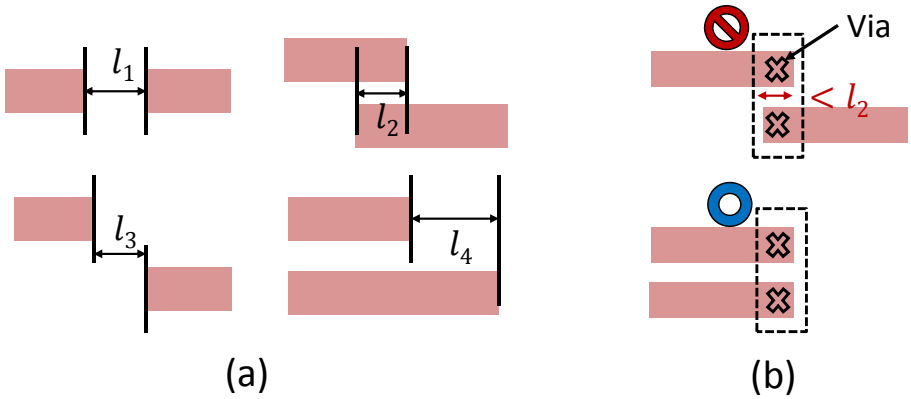


Figure 4.3: (a) SADP-specific design rules. (b) Example showing that via location does not provide enough information to distinguish the upper and lower cases, i.e., to check SADP-aware rules.

In our ILP formulation, we use via locations to check the locations of EOL, but this is not enough to differentiate the two cases in Figure 4.3(b), where the upper case is an illegal routing while the lower case is legal with the same via placements. Therefore, binary variables $p^k_{r,i}$, $p^k_{l,i}$ that indicate whether there are flows connected to a vertex v_i that come from right or left direction, respectively, are defined for a net n_k to represent the directions of the EOL. Note that there are only two directions, since we assume unidirectional routing.

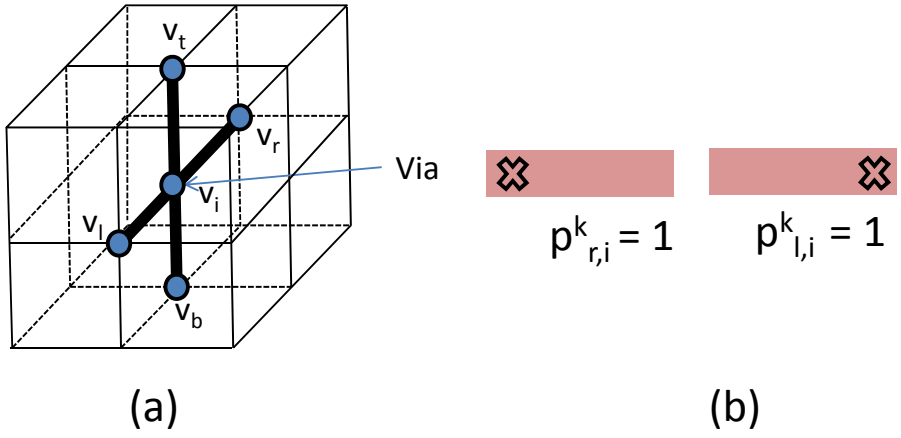


Figure 4.4: An example of a routing graph. (a) The p variable of a vertex v_i is determined by flow variables of edges with vertex v_i's neighbor vertices v_t, v_b, v_l and v_r. (b) Wire segment geometries that respectively result when $p^k_{r,i} = 1$ and $p^k_{l,i} = 1$.

Figure 4.4(a) shows a vertex v_i and its top, bottom, left, right neighbor vertices (v_t, v_b, v_l, v_r) in a routing graph, and Figure 4.4(b) enumerates the cases when each p variable $= 1$. For the left EOL at (x_i,

y_i, z_i) in Figure 4.4(a), where $p^k_{r,i} = 1$, the right edge ($e^k_{r,i}$) connected to v_i must be used in routing and the left edge ($e^k_{l,i}$) connected to v_i must not be used. By Constraint (4.4), the expression $e^k_{r,i}$ && $\neg e^k_{l,i}$ is equal to the right-hand side of Constraint (4.7). The right EOL ($p^k_{l,i} = 1$) is formulated in the same manner, i.e., as Constraint (4.6).

$$p^k_{l,i} = (e^k_{l,i} * e^k_{i,t}) || (e^k_{l,i} * e^k_{i,b}) || (e^k_{i,l} * e^k_{t,i}) || (e^k_{i,l} * e^k_{b,i}) \tag{4.6}$$

$$p^k_{r,i} = (e^k_{r,i} * e^k_{i,t}) || (e^k_{r,i} * e^k_{i,b}) || (e^k_{i,r} * e^k_{t,i}) || (e^k_{i,r} * e^k_{b,i}) \tag{4.7}$$

As all the variables are binary, we can convert the quadratic constraints in Constraint (4.6) and (4.7) to linear constraints by using a simple technique as shown in (4.8). The $(a \leq b)$ && $(a \leq c)$ condition ensures that a is zero when either b or c is zero. The condition $(a \geq b + c - 1)$ makes $a = 1$ when both b and c are one.

$$a = b * c \Longleftrightarrow (a \leq b) \;\&\&\; (a \leq c) \;\&\&\; (a \geq b + c - 1) \tag{4.8}$$

We then convert the Constraint (4.6) to a set of linear constraints as shown in Constraint (4.9).

$$\begin{aligned}
p^k_{l,i} \geq p^k_{l,i,1} \;;\; p^k_{l,i} \geq p^k_{l,i,2} \;;\; p^k_{l,i} \geq p^k_{l,i,3} \;;\; p^k_{l,i} \geq p^k_{l,i,4} \\
p^k_{l,i} \leq p^k_{l,i,1} + p^k_{l,i,2} + p^k_{l,i,3} + p^k_{l,i,4} \\
(p^k_{l,i,1} \leq e^k_{l,i}) \;\&\&\; (p^k_{l,i,1} \leq e^k_{i,t}) \;\&\&\; (p^k_{l,i,1} \geq e^k_{l,i} + e^k_{i,t} - 1) \\
(p^k_{l,i,2} \leq e^k_{l,i}) \;\&\&\; (p^k_{l,i,2} \leq e^k_{i,b}) \;\&\&\; (p^k_{l,i,2} \geq e^k_{l,i} + e^k_{i,b} - 1) \\
(p^k_{l,i,3} \leq e^k_{i,l}) \;\&\&\; (p^k_{l,i,3} \leq e^k_{t,i}) \;\&\&\; (p^k_{l,i,3} \geq e^k_{i,l} + e^k_{t,i} - 1) \\
(p^k_{l,i,4} \leq e^k_{i,l}) \;\&\&\; (p^k_{l,i,4} \leq e^k_{b,i}) \;\&\&\; (p^k_{l,i,4} \geq e^k_{i,l} + e^k_{b,i} - 1)
\end{aligned} \tag{4.9}$$

Here, $p^k_{*,*}$ is a net-specific variable. As SADP rules must be checked over all nets, we define global p variables as follows:

$$p_{l,i} = \sum_{n_k \in K} p^k_{l,i}, \;\; p_{r,i} = \sum_{n_k \in K} p^k_{r,i} \tag{4.10}$$

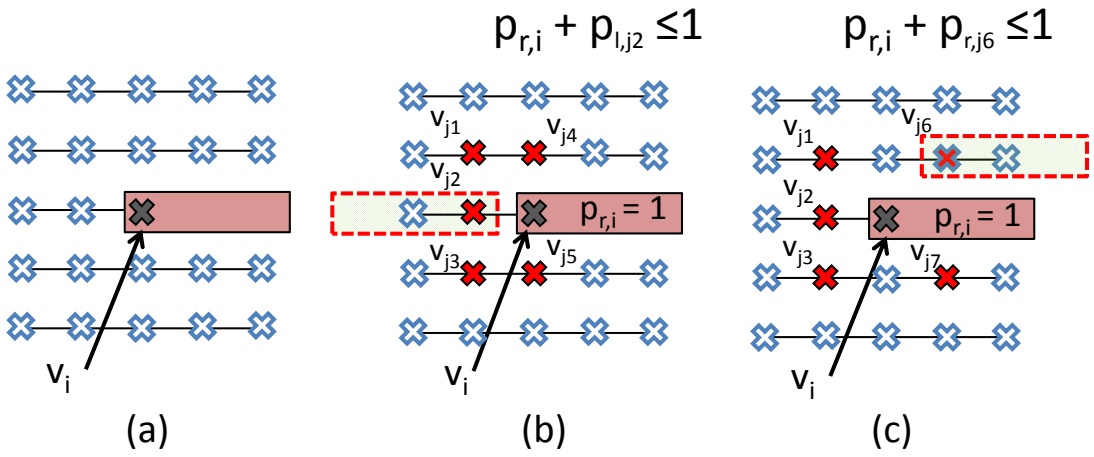


Figure 4.5: (a) A wire segment, of which the EOL is located at vertex v_i with the wire coming from the right side. (b) Forbidden via locations for other wire segments with $p_{l,j} = 1$. (c) Forbidden via locations for other wire segments with $p_{r,j} = 1$.

Figure 4.5 shows how the p variables can be used to formulate SADP-aware rules for ILP. Figure 4.5(a) shows a wire segment, of which the EOL is located at vertex v_i with the wire coming from the right side; (b) and (c) show forbidden via locations for the other wire segments. The constraints shown in Figures 4.5(b) and (c) are formulated as Constraints (4.11) and (4.12), respectively.

$$\begin{aligned}(p_{r,i} + p_{l,j_1} \leq 1) \;\&\&\; (p_{r,i} + p_{l,j_2} \leq 1) \;\&\&\; (p_{r,i} + p_{l,j_3} \leq 1) \\ \&\&\; (p_{r,i} + p_{l,j_4} \leq 1) \;\&\&\; (p_{r,i} + p_{l,j_5} \leq 1)\end{aligned} \tag{4.11}$$

$$\begin{aligned}(p_{r,i} + p_{r,j_1} \leq 1) \;\&\&\; (p_{r,i} + p_{r,j_2} \leq 1) \;\&\&\; (p_{r,i} + p_{r,j_3} \leq 1) \\ \&\&\; (p_{r,i} + p_{r,j_6} \leq 1) \;\&\&\; (p_{r,i} + p_{r,j_7} \leq 1)\end{aligned} \tag{4.12}$$

4.1.3 Empirical Studies

Our empirical studies seek to answer two basic questions:

- What are the design costs of various BEOL rules with respect to wirelength, the number of vias, and routability metrics?
- How much do impacts of design rules vary across different technologies and different-track cell architectures?

Overall flow of BEOL rule evaluation. We implement our experimental testbed in C++ code, with interface to support LEF/DEF [186] implemented via the *OpenAccess 2.6* [193] API. We use *IBM ILOG CPLEX Optimization Studio v12.5.1* [180] as our ILP solver. Figure 4.6 shows our overall BEOL rule evaluation flow. From a routed design, all possible routing clips are extracted, and evaluated according to our pin cost metric. The clips with highest pin cost are selected, and each clip (= switchbox instance) is converted to a routing graph based on available metal tracks, and then to a corresponding ILP instance, for each routing rule configuration that we study. Our *OptRouter* then obtains optimal routing solutions by solving the ILP. From the solution, we report out wirelength, number of vias, and feasibility (routability) for each given clip with each given rule configuration. For the experiments that we report here, routing cost in the ILP is defined as *wirelength* + $4\times$ *number of vias*. We have separately observed that the ILP sensibly handles alternative routing cost definitions with different weighting of via count.

Physical implementation with advanced technology. We verify our methods using the open-source AES design [191] and an ARM Cortex M0, implemented with three different technologies and standard-cell libraries: 8-track in $28nm$ FDSOI (N28-8T), 12-track in $28nm$ FDSOI (N28-12T), and 9-track in $7nm$ (N7-9T). We use *Synopsys Design Compiler H-2013.03-SP3* [195] for synthesis and *Cadence Encounter Digital Implementation System 13.1* [171] for P&R. We implement each design multiple times, with a range of final utilizations. Table 4.2 summarizes benchmark design information.

For $7nm$ technology, we use $7nm$ standard-cell libraries (P&R, layout and timing views) from a leading IP provider; metal pitches on layers M1 to M6 and layers M7 to M8 are $40nm$ and $80nm$,

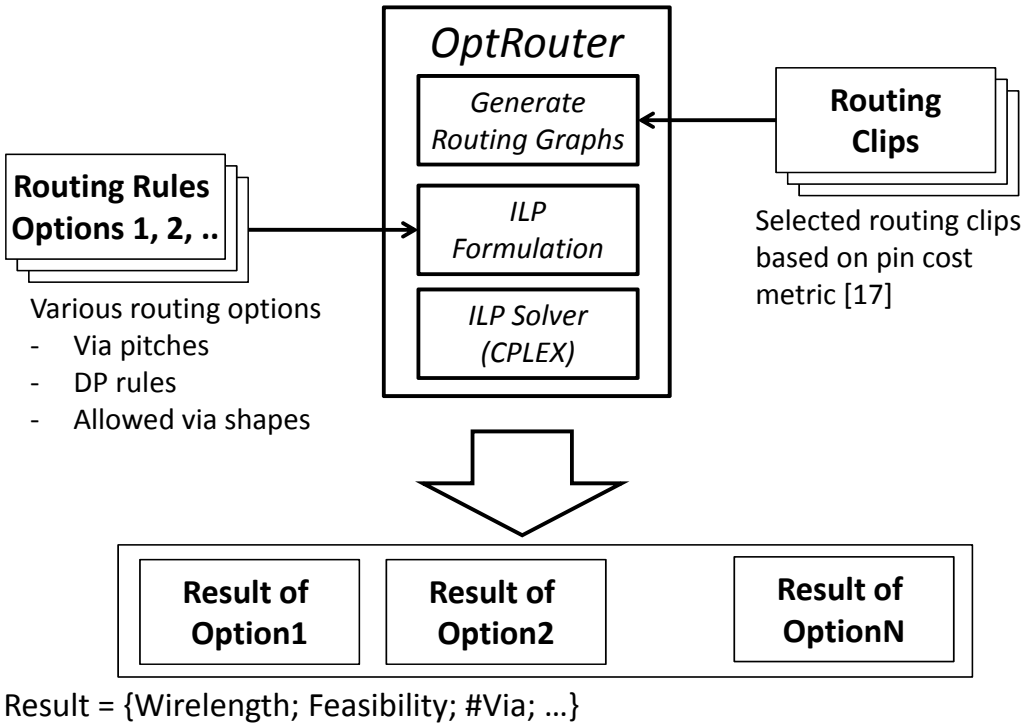


Figure 4.6: Overall flow of BEOL rule evaluation.

Table 4.2: Benchmark designs.

Tech.	Design	Period (ns)	#Inst.	Util. (%)
N28-12T	AES	1.2	13.5–14K	89–94
	M0	2.2	9.2K	90–96
N28-8T	AES	2	12–12.7K	89–95
	M0	2.5	9.3–9.5K	90–95
N7-9T	AES	0.6	13–15K	93–97
	M0	1.2	9.7–11.4K	92–95

respectively. In this technology, our design enablement is missing detailed BEOL technology information such as RC values and BEOL stack options. Thus, to obtain timing-closed P&R results we scale up the geometries of the $7nm$ 9-track cells by 2.5$\times$ in the vertical dimension (i.e., by the ratio of 1$\times$ metal pitch in $28nm$ horizontal layers ($100nm$) to 1$\times$ metal pitch in $7nm$ horizontal layers ($40nm$)). Then, the scaled $7nm$ cells fit into the $28nm$ BEOL stack with the same number of horizontal metal tracks, for which we use $100nm$ metal pitch in horizontal layers. To scale the widths of the $7nm$ standard cells, we scale by the ratio of the $28nm$ placement grid (vertical metal layer pitch of $136nm$) to that of the $7nm$ placement grid (vertical metal layer pitch of $54nm$), which is $\sim$2.5. We further adjust pin locations so that pins are

on-grid, since simple scaling results in off-grid pins which affect routability.[29] To derive the missing $7nm$ wire RC information from $28nm$ RC values, we scale up R by $15\times$ for $7nm$ wire R, and use the same wire C value. This follows methodology of, e.g., [23] to account for the rapid increase of resistivity in advanced nodes. Then, since we are using the scaled geometries to mimic a $7nm$ P&R flow, R and C per unit length are scaled down (in the P&R tool) by $2.5\times$. The end result is that $7nm$ R and C per unit length (R_{N7}, C_{N7}) are obtained from $28nm$ R and C per unit length (R_{N28}, C_{N28}) as $R_{N7} = 6 \times R_{N28}$ and $C_{N7} = C_{N28}/2.5$.

Extraction of routing clips. We use 1μm $\times$ 1μm routing clips extracted from the routed designs as input instances; these correspond to 7 vertical routing tracks $\times$ 10 horizontal routing tracks, with eight metal layers, for *OptRouter*. Figure 4.7 shows example routing clips extracted from layouts with (a) N28-12T cells, (b) N28-8T cells, and (c) N7-9T cells.[30]

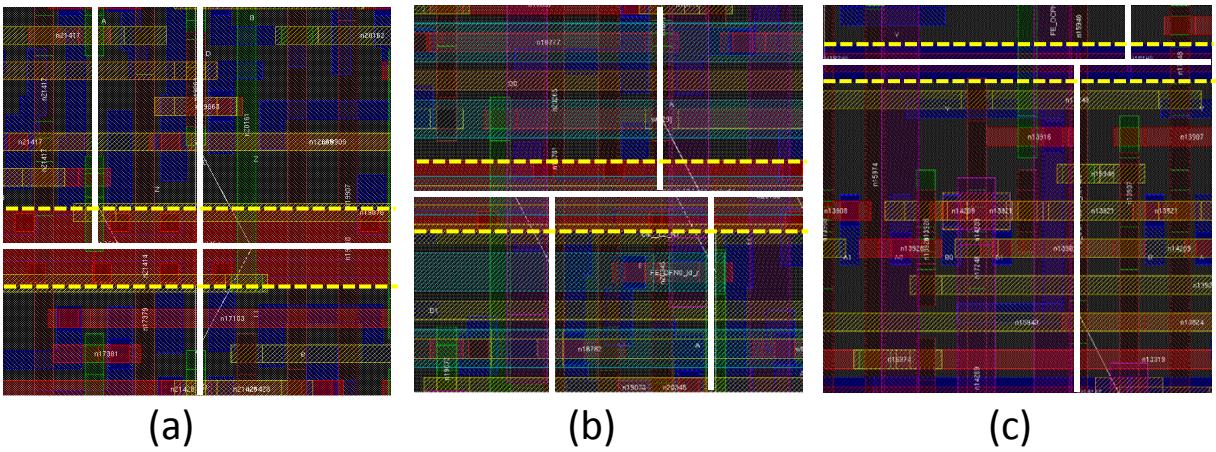


Figure 4.7: Routing clips from (a) N28-12T, (b) N28-9T and (c) N7-9T. Standard cell boundaries and power/ground rail are highlighted with white lines and yellow dashed lines, respectively.

We select "difficult-to-route" clips based on pin cost metrics of Taghavi et al. [148], specifically, a pin existence cost (PEC), a pin-area cost ($PAC = \sum_{i=1}^{PEC} 2^{2-\frac{area(p_i)}{\theta}}$) and a pin-spacing cost ($PRC =$

[29] In greater detail: the $28nm$ and $7nm$ placement grids are $136nm$ and $54nm$, respectively, with ratio between the two being $\sim$2.519. It is not possible to obtain integer cell widths by simply scaling with this number. Thus, we scale up the $7nm$ cells by 2.5 so that all cell widths are a multiple of $135nm$. We then increase each cell width by *scaled cell width* $/135$ in order to make it a multiple of $136nm$, which is the foundry $28nm$ placement grid. Since scaling by $2.5\times$ results in a pin pitch of $135nm$, which is off-grid with respect to a $136nm$ grid, we perform a scripted movement of pin locations so that all pins are again on-grid (the pin x locations should be multiples of $136nm$).

[30] By comparison, the recent work of [78] uses 1.26μm $\times$ 1.26μm clips in a $45nm$ technology; these correspond to 9 vertical routing tracks $\times$ 9 horizontal routing tracks.

$\sum_{i=1}^{PEC-1} \sum_{j=i+1}^{PEC} 2^{2-\frac{spacing(p_i,p_j)}{3\theta}}$). We use $PEC + PAC + PRC$ as the *pin cost* for a routing clip, with $\theta = 500$ to obtain a reasonable range of costs.

We calculate the pin cost for every routing clip in the routed testcases listed in Table 4.2 ($\sim$10K clips per testcase). Figure 4.8 shows the top-100 pin cost ranges for several versions of AES and M0 design implementations in N7-9T with different utilizations. The utilizations of AES_v1, AES_v2 and AES_v3 are 93%, 95% and 97% respectively, and the utilizations of M0_v1, M0_v2 and M0_v3 are 92%, 94% and 95%[31]. We observe that pin cost distributions do not change significantly with different utilizations, and that pin cost distributions are not design-specific: ranges of top-100 pin costs of both designs are similar (AES: 33$\sim$42, M0: 30$\sim$41). Thus, in each technology we select top-100 clips from across all design implementations, according to the pin cost metric.

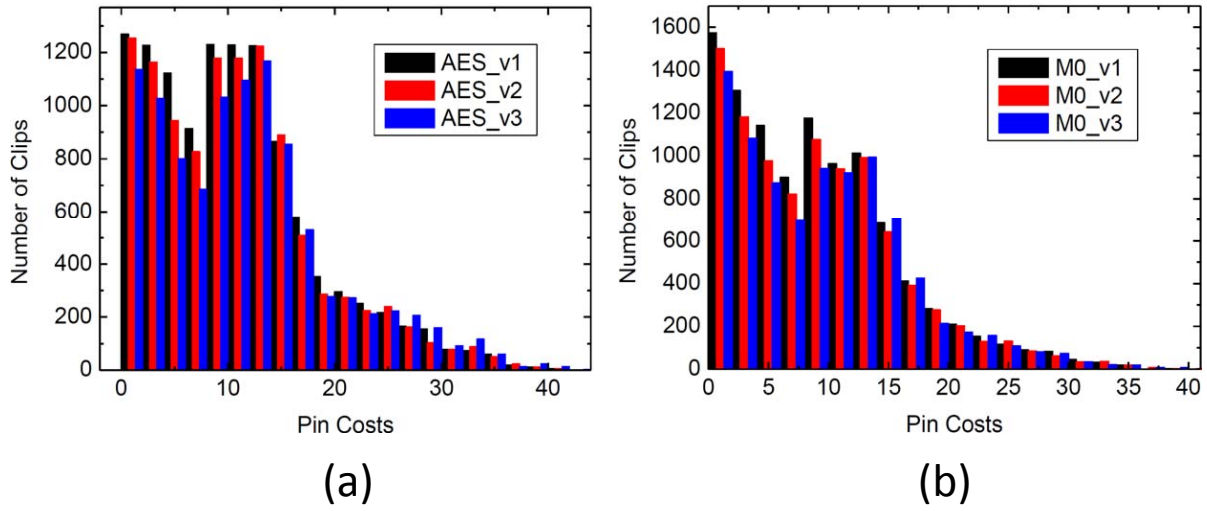


Figure 4.8: Pin cost distributions (per the *PEC + PAC + PRC* metrics in [148]) of (a) AES and (b) M0 with different utilizations.

Design of Experiments

We evaluate various BEOL design rule configurations, each of which is a combination of via restrictions and mix of LELE/SADP BEOL layers. (All routing layers are unidirectional in our study.) Table 4.3 shows the BEOL design rule configurations, denoted as RULE1-RULE11, used in the experiments.

[31] We use high utilizations to obtain designs that are "difficult-to-route" and sensitive to design rules due to routing congestion.

We test three via restriction cases (*0 neighbors blocked*; *4 neighbors blocked*; and *8 neighbors blocked*) and five LELE/SADP layer combinations (M2-M8 LELE layers (*No SADP*); M2-M8 SADP layers (*SADP* $\geq$ *M2*); M2 LELE + M3-M8 SADP layers (*SADP* $\geq$ *M3*); M2-M3 LELE + M4-M8 SADP layers (*SADP* $\geq$ *M4*); and M2-M4 LELE + M5-M8 SADP (*SADP* $\geq$ *M5*)). Via restrictions are applied to the V12 through V78 layers.

We select the top 100 routing clips according to pin costs across all designs in Table 4.2, for each of the three combinations of technology node and cell height, as discussed above. We then run *OptRouter* on each of the 100 routing clips to evaluate the impact of each given routing rule configuration. We obtain the Δcost of each rule configuration, relative to the routing cost of RULE1 (no constraints).[32] In the present study, we do not use M1 as a routing resource.

Table 4.3: BEOL design rule configurations.

Name	SADP rules	Blocked via sites
RULE1	No SADP	0 neighbors blocked
RULE2, RULE3, RULE4, RULE5	SADP $\geq$ {M2, M3, M4, M5}	
RULE6	No SADP	4 neighbors blocked
RULE7, RULE8	SADP $\geq$ {M2, M3}	
RULE9	No SADP	8 neighbors blocked
RULE10, RULE11	SADP $\geq$ {M2, M3}	

We have evaluated all of RULE1 to RULE11 for the N28-12T and N28-8T technologies. However, we do not test RULE2, RULE7, and RULE9 to RULE11 for N7-9T since the smaller pin shapes in the $7nm$ standard cells do not permit the diagonal (adjacency in) via placement which is required for these rules. Figures 4.9(a), (b) and (c) show pin shapes in a NAND2X1 cell in N28-12T, N28-8T and N7-9T, respectively. In Figure 4.9(c), the input pin shapes have only two access points and the two pins are close to each other. With eight via sites blocked, it is impossible to connect to the two input pins without violations.

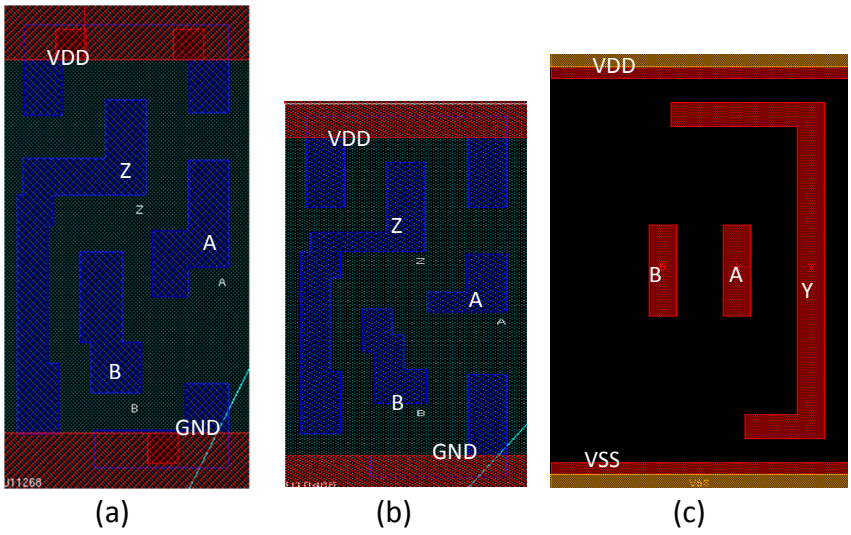


Figure 4.9: Pin shapes in NAND2X1: (a) N28-12T, (b) N28-8T and (c) scaled N7-9T.

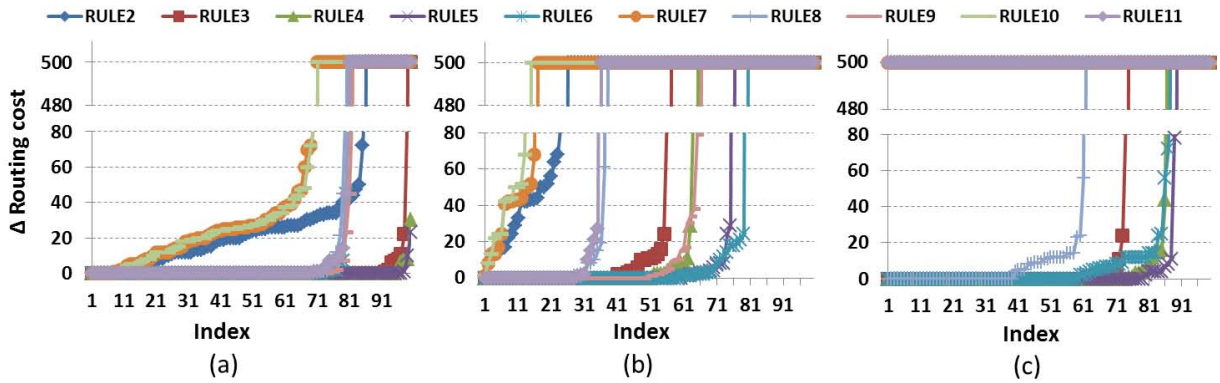


Figure 4.10: Δcost with different RULE* in (a) N28-12T, (b) N28-8T and (c) N7-9T.

Experimental Results

Figures 4.10(a), (b) and (c) respectively show sorted Δcost per clip of each RULE, in N28-12T, N28-8T and N7-9T cell-based designs. The Δ is relative to costs with RULE1 (i.e., the minimum achievable routing cost with eight unidirectional LELE layers and no via restrictions). For unroutable clips, we arbitrarily set Δcost=500 for convenience of plot generation.

In N28-12T (Figure 4.10(a)), we observe that SADP rules for upper metal layers above M3 do not

[32]Our separate studies support the claimed optimality of OptRouter. We have compared the results of OptRouter and those of the commercial routing tool, and have found that OptRouter always achieves non-positive Δcost with respect to the commercial tool's solution. Indeed, the average Δcost of -10$\sim$-15, relative to an average routing cost of $\sim$380, suggests the potential for using OptRouter for detailed routing improvement.

significantly affect routing costs. Two kinds of via restrictions (4 or 8 neighbors blocked) show similar routing costs, suggesting that the orthogonal via restriction (4 neighbors blocked) is dominant. When SADP rules are applied (RULE4, RULE7, RULE2), routing costs vary across routing clips. By comparing the "crossing" traces for RULE2 and RULE6, we see that routing costs are higher with SADP layers than with via restriction rules, but that in terms of absolute feasibility, the via restriction appears to result in fewer feasible routings.

With N28-8T (Figure 4.10(b)), in contrast to the N28-12T case, there is higher sensitivity of Δcost to the number of SADP layers: we see a clear increasing cost trend across RULE2 through RULE5. With respect to via restriction, for the 8 LELE layer (no SADP) cases, having 4 and 8 neighbors blocked yields different pin cost distributions (RULE6 versus RULE9), i.e., the orthogonal via restriction is less dominant in this particular context. However, when via restriction is combined with SADP layers, the two forms of via restriction again show similar results (RULE7 versus RULE10, RULE8 versus RULE11).

With N7-9T (Figure 4.10(c)), SADP routing rules have less cost impact on layers above M4, as RULE4, RULE5 and RULE6 show similar Δcost distributions. When M3 is made into an SADP layer (RULE3), the vertical line (i.e., the clip index at which sorted Δcost goes to infinity (infeasible solution)) shifts left significantly. When the 4-neighbors via restriction is added to RULE3 (i.e., in RULE8), the vertical line shifts again. (RULE3, RULE4, RULE5, RULE6 and RULE8 respectively have 26, 14, 11, 13 and 39 infeasible clips out of 100.)

From the preceding discussions, we may tentatively form two general observations. (1) First, the via restriction and SADP routing rules show different trends, i.e., effects on the Δcost profile. Moreover, the sensitivities of Δcost to design rules and routing options vary with technology. For example, when SADP rules are applied to upper metal layers in N28-12T or N7-9T, the routing costs do not change significantly, which we interpret to mean that SADP rules do not affect routability significantly for these clips. This is different from what we observe in N28-8T. (2) Second, for design rules that are applied to upper metal layers (>M3), almost half of routing clips show zero Δcost. This could imply that the pin cost metric of [148] cannot, by itself, accurately quantify the difficulty. In other words, there is a gap between pin accessibility metrics such as [148] and our switchbox-centric evaluation of routability.

Analysis of the number of variables and constraints. The number of directed arcs ($|A|$), the

number of vertices ($|V|$) and the number of nets ($|N|$) determine the number of variables and constraints in the ILP. Without via restrictions and SADP rules (no restriction), the number of variables is $O(|A| \cdot |N|)$. With Constraints (4.1)–(4.4), the number of constraints is $O((|V| + 3 \cdot |A|) \cdot |N|)$. Regarding via restriction, when α neighbor sites are blocked, the number of variable is the same as the basic case (no restriction), and the number of constraints is $O(\alpha \cdot |V| + (|V| + 3 \cdot |A|) \cdot |N|)$. With the SADP routing rules, the number of variables is $O((10 \cdot |V| + |A|) \cdot |N|)$ because of the additional binary indicator (p); the number of constraints is $O((34 \cdot |V| + 3 \cdot |A|) \cdot |N| + 10 \cdot |V|)$. Regarding via shapes, when a β size of via shape is considered, the number of variables is $O((\beta \cdot |V| + |A|) \cdot |N|)$ due to the creation of additional via edges, and the number of constraints is $O(\beta^2 \cdot |V| \cdot |N| + (\beta \cdot |V| + 3 \cdot |A|) \cdot |N|)$.

4.1.4 Conclusion

In this work, we have studied impacts of patterning technology choices and design rules on physical implementation metrics, with respect to *cost-optimal* design rule-correct detailed routing. We describe *OptRouter*, an ILP-based optimal detailed router that correctly handles multi-pin nets and various sub-$20nm$ routing challenges including via restrictions, via shapes, and SADP patterning rules. OptRouter enables design rule evaluation using "difficult" routing clips (switchboxes) selected according to a pin cost metric. We study Δcost distributions for different design rules, relative to a RULE1 where all layers are LELE and there are no via restrictions. From the results, we observe that the sensitivities of Δcost to design rules and routing options vary with technology. Also, we observe that there is a gap between pin accessibility metrics such as [148] and our switchbox-centric evaluation of routability.

Future work includes speedup of OptRouter to gain insights into physical implementation impacts at larger granularity (switchbox size). Currently, OptRouter average runtime for a 7 track $\times$ 10 track switchbox ($1.0 \times 1.0\mu m^2$ layout area in $28nm$) is 1047 seconds (single-threaded) with SADP and via restriction rules. Without such rules (as in [78]), average runtime is 842 seconds.[33] As noted above, our results give insight into the degree of suboptimality in current routing tools, and open up the possibility of (massively distributed) local improvement of detailed routing solutions. Also, for better quantification of

[33]OptRouter runtime for a 10 track $\times$ 10 track switchbox, with (resp. without) SADP and via restriction rules, is 1340 (resp. 925) seconds.

"difficult-to-route" clips, development of a metric beyond [148] to estimate routability in sub-20nm nodes will be an important aspect of our future work.

4.2 Performance- and Energy-Aware Optimization of BEOL Interconnect Stack Geometry in Advanced Technology Nodes

In advanced technology nodes, power and performance requirements are increasingly stringent even as classical Moore's-Law scaling has slowed down. BEOL interconnect stack geometry has become a key lever for design enablement. Reasons for this include: (i) the resistance of Cu interconnect has increased dramatically in sub-100nm nodes due to grain boundary and trench liner effects [79]; and (ii) the scaling of effective dielectric constant has slowed in recent years, resulting in severely increased interconnect capacitance [182] and diminished performance benefits at new nodes. The resulting rapid increase of interconnect RC leads to not only performance loss from interconnect delay increase, but circuit power and area degradation as well. In this work, we study the potential value of BEOL interconnect stack geometry optimizations, by exploring wire aspect ratio (AR) and wire line-space duty cycle (DC). Our broad objective is to assess whether new manufacturing-aware design (MAD) [54] and design-aware manufacturing (DAM) methodologies can contribute "equivalent scaling" in the N7/N5 nodes and beyond.

Industry Implementations

For interconnect geometry optimization, wire height/width aspect ratio (AR) and wire width/pitch duty cycle (DC) are the two obvious levers for a given metal pitch value and BEOL process (see Figure 4.11). In the most recent technology nodes, IC companies have deviated from "classic" 2:1 AR and 50% DC for each metal layer, for reasons of performance, energy, reliability and manufacturability. Narasimha et al. [118] achieve 20% reduction in RC delay by optimizing liner resistivity and metal line aspect ratio of 1$\times$ layers in IBM's high-performance 45nm SOI technology node. Jan et al. [76] describe two different interconnect geometries that meet different product types, power and performance goals in Intel's 22nm node. Figure 4.12 depicts the BEOL stacks for high-performance CPU and high-density SoC in Intel's 22nm node [76]. For high-performance CPU, thicker and wider wires with large AR and DC are observed.

By contrast, flat wires with moderate DC are used for high-density SoC. Zhu et al. [167] have patented a local optimization to improve SoC performance using two BEOL stacks. The first stack is used for non-critical blocks while the second stack, with larger line width and via width, is used for critical blocks.

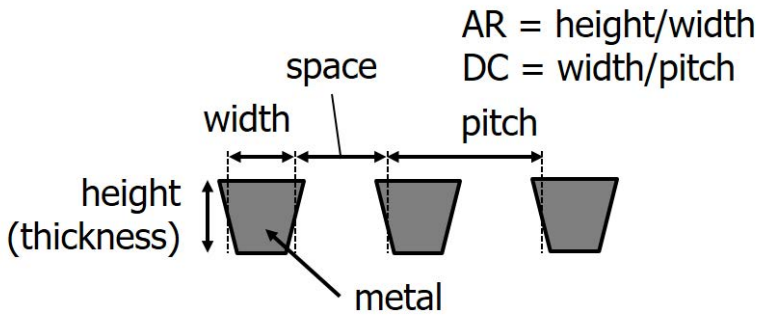


Figure 4.11: Illustration of height/width aspect ratio (AR) and width/pitch duty cycle (DC).

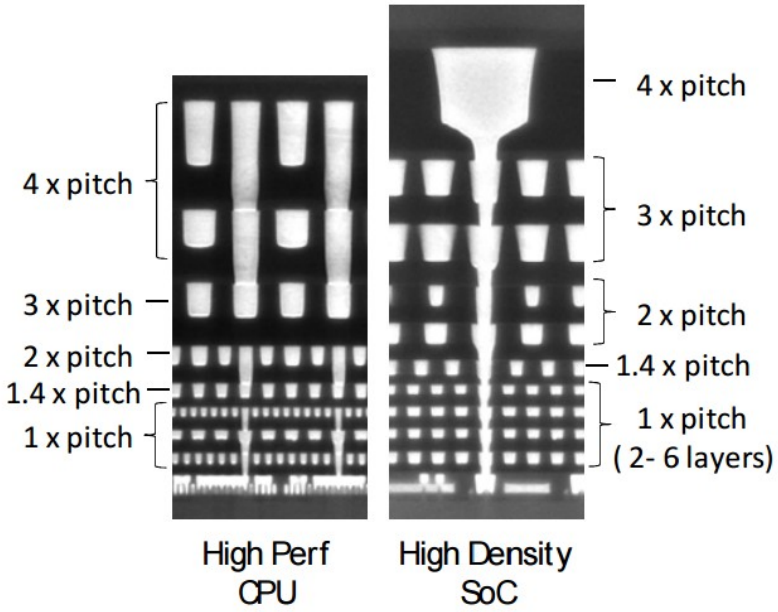


Figure 4.12: Interconnect architecture comparison of $22nm$ CPU and SoC [76].

Current Approaches and Limitations

Even as sophisticated IC companies have adopted various choices of AR and DC for different design targets in each technology node, to our knowledge it is not obvious how to optimize BEOL dimensions, and there is no general methodology to identify an "optimal" BEOL stack option for a given

design. Two levels of optimization exist in previous works: (i) device level, and (ii) block level. For (i), the Elmore delay model [151] provides fast modeling of RC networks. Elfadel et al. [40] describe AQUAIA, which enables fast modeling and simulation of delay, slew and crosstalk. Faruk et al. [41] utilize AQUAIA for variability modeling with different line widths, heights, pitches and dielectric constants. Bakoglu et al. [12], Ismail et al. [74] and Pamanuwa et al. [123] develop delay models considering repeater insertion, inductance and coupling capacitance. These techniques enable fast modeling of delay and energy for given driver, load and interconnect structures. Thus, for instance, "optimal" single-stage (AR,DC) with fixed driver and load can be determined by sweeping (AR,DC) combinations. For (ii), Anand et al. [9][8] develop a framework to optimize a metal stack in a more global sense. The authors conclude with a suggestion of low AR and DC. Takahashi et al. [149] propose a methodology for determining overall interconnect strategy, including repeater insertion, as well as adoption of new metal and dielectric materials. Other works including [168] optimize DC only, for long interconnects. Recently, [140] performs block-level validations of BEOL optimization based on results from single-stage simulation for advanced nodes. The work of [30] suggests that different optimal (AR,DC) combinations may apply when considering driver and load.

Our Approach

In this work, we study optimization of BEOL interconnect stack geometry through exploration of wire aspect ratio (AR) and wire width/pitch duty cycle (DC) impacts in sub-10nm nodes. We perform SPICE-based analyses of timing path delays to find delay- or power-optimal (AR,DC) combinations, and also perform block-level studies with placed and routed designs. Based on our experimental results, we provide various insights on BEOL stack geometry: (i) optimal (AR,DC) for a given wire pitch with respect to power and delay; (ii) sensitivities of optimal (AR,DC) to circuit parameters (e.g., driver strength, input slew, output load, wirelength); (iii) block-level optimal (AR,DC) with multiple interconnect layers; and (iv) potential impacts of future design-aware manufacturing (DAM) and/or manufacturing-aware design (MAD) methodologies [54] that co-optimize product designs and BEOL interconnect stacks.

The contributions of this work are summarized as follows.

- We explore various wire dimensions using SPICE simulation and determine "optimal" wire dimen-

sions (AR,DC) for a given metal pitch with respect to power and performance.

- We study the sensitivities of (AR,DC) optimization to several parameters that determine circuit performance and power (e.g., driver strength, input slew, output load, wirelength).

- We show that performance and power results from a standard place-and-route (P&R) flow (including post-route parasitic RC extraction (PEX) and static timing analysis (STA)) are consistent with our SPICE simulation-based results.

- We investigate the potential impacts of future design-aware manufacturing (DAM) and manufacturing-aware design (MAD) methodologies by performing P&R with real block-level designs.

4.2.1 Related Work

In this section, we review two important related works on BEOL interconnect optimization for sub-14nm nodes.

Shah [140] analyzes the impact of local layer interconnect dimensions on the performance of single-stage, single-size inverter circuits for various technology nodes, and identifies optimal AR and DC values with respect to slew-bounded delay and slew-bounded energy-delay product (EDP). Based on SPICE simulation and field-solver analyses for single gate-interconnect stages, it is shown that the combination of low AR and high DC can achieve a better overall performance for advanced nodes. The author provides validations using predictive technology models, and furthermore studies multi-stage impact using a physical design flow for sample benchmark designs and a random path model. While [140] considers effects seen in advanced process technologies, such as new barrier and dielectric materials and the impact of process variations based on the ITRS roadmap [182], several limitations are noted. First, predictive technology models and scaled libraries for advanced nodes are based on generic planar-bulk 32/28nm libraries, with mismatched scaling of parasitics versus the dimensions of devices and interconnects; this may not accurately match current and impending 7nm/5nm FinFET nodes. Second, optimal wire dimensions are determined only for local metal layers, whereas long interconnects on higher layers may play a more important role in BEOL interconnect optimization. Third, the design-level analysis of [140] only compares power and timing performance implications of the suggested optimal AR and DC values to those of ITRS

predicted values. Tradeoffs of AR and DC at the design level are not contemplated in [140], as the work focuses only on single-stage analysis for fixed-size inverters.

Ciofi et al. [30] investigate the impact of wire geometry on the resistance, capacitance, and RC delay of Cu/low-k damascene interconnects for fixed line-to-line pitch for the $7nm$ logic technology node. The resistance is computed by applying a semiempirical resistivity model and the capacitance is simulated by means of a 2D field solver. The authors show that RC delay can be significantly reduced by trading capacitance for resistance with wider and thicker wires. They also show that a given RC delay can be achieved with several geometries, which provides a useful degree of freedom for system-level optimization. Next, the authors suggest that the optimal point for circuit performance in terms of power and delay may differ from the RC delay and give delay and power contours for different AR and DC combinations for $\times 1$ and $\times 4$ drivers with a wirelength of 300 contacted poly pitches (CPPs). However, this work does not suggest any method to incorporate the existing findings to block-level designs, which include different types of cells, and semi-global/global interconnect layers.

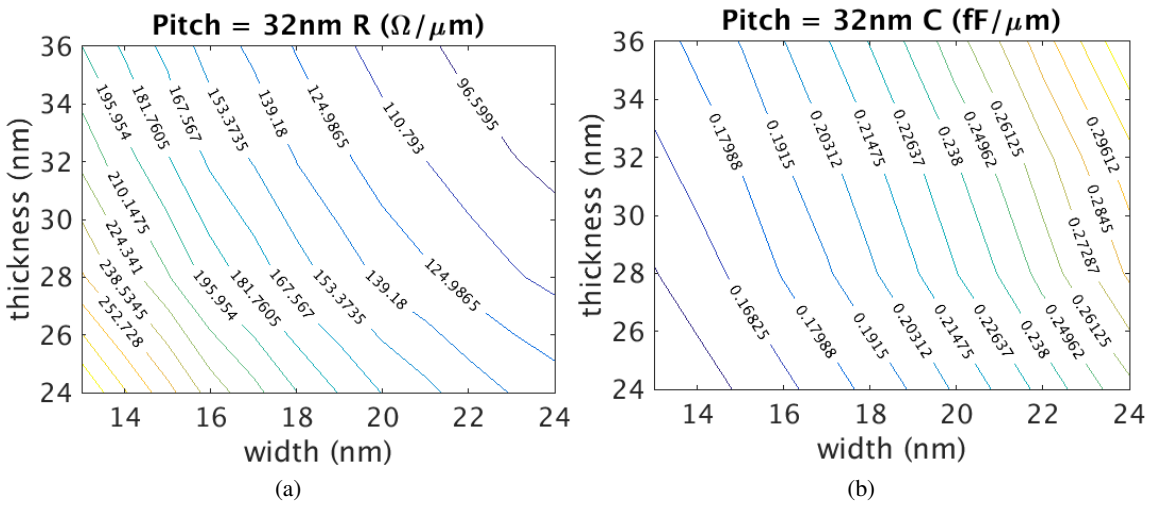
4.2.2 Path-Based Simulation

We now describe our methodology to evaluate power and delay impacts of various BEOL interconnect stack geometries, based on path-based SPICE-level simulations. Based on SPICE-level simulation, we study the sensitivities of delay and power to driver strength, wirelength, output load and input slew. We further show that the P&R flow's analysis results (i.e., including PEX and STA) are well-correlated with SPICE-level simulation results.

Single-Stage SPICE Simulation

For SPICE-level simulation, we evaluate the power and delay impacts of various wire dimensions using single-stage circuits. We first extract RC values per unit length (μm) with different AR and DC values for three metal layer types[34] and use these values to construct single-stage circuits with various configurations (i.e., sizes of buffers, wirelength, output load and input slew).

[34] We consider $1\times$, $1.5\times$ and $2.5\times$ layers (i.e., pitch = 32, 48 and $80nm$).



(a) (b)

Figure 4.13: Contour maps of (a) resistance and (b) capacitance per unit length (μm) for metal pitch $32nm$.

RC extraction for various wire dimensions. We perform parasitic RC extraction using *Cadence QRC* [175] with QRC techfiles and LEF files [186] to obtain per-unit length (μm) R and C values for various wire dimensions. To model different wire thicknesses, we generate multiple QRC techfiles with ICT [175] files that are modified with various thickness values for each metal layer, using *Cadence QRC Techgen* [175]. To sweep metal width values, we modify the "WIDTH" and "SPACING" fields in LEF files.[35] To obtain wire RC with fine-grained width and thickness values, we perform linear interpolation. Figures 4.13(a) and (b) show the contour maps of per-unit length (μm) $\bar{r}$ and $\bar{c}$ for metal pitch $32nm$[36], respectively. Both $\bar{r}$ and $\bar{c}$ increase with larger width and thickness values, as expected. Our observations are consistent with those reported in [30].

Circuit structure for SPICE simulation. Figure 4.14 shows the circuit structure that we use for SPICE simulation. With the extracted per-unit length (μm) values $\bar{r}$ and $\bar{c}$, we compute wire resistance R_{wire} and capacitance C_{wire} for a given wirelength. We then construct an RC circuit using the $\Pi 3$ model for wire segments.

Sensitivity of power and delay to input configurations. Using SPICE simulation with the circuit structure described above, we study the sensitivities of performance and power to several parameters with

[35]The default ICT and LEF files that we use are provided by our collaborators at a leading technology consortium. In our study, we do not investigate patterning or manufacturing issues that may pertain to different (AR,DC) combinations.

[36]As in [30], we focus on $1\times$ metal layers in N7/N5 nodes.

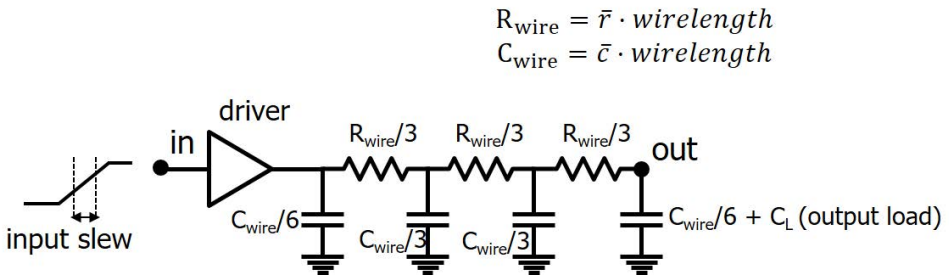


Figure 4.14: Circuit structure for SPICE simulation.

$1\times$ metal layer (pitch = $32nm$). We vary driver strength, wirelength, output load and input slew, as follows. The values in bold font are defaults.

- driver strength = {X1, **X4**, X8, X16}
- wirelength = {$5\mu m$, **10**μm, $15\mu m$, $20\mu m$}
- output load = {$2fF$, $3fF$, **5**fF, $10fF$}
- input slew = {**50**ps, $100ps$}

Figures 4.15(a), (b), (c) and (d) show the power and delay contour maps for various driver sizes, i.e., BUF_X1, BUF_X2, BUF_X8 and BUF_X16, respectively. We observe that (i) with BUF_X1, smaller width and thickness values are always better for both power and delay (no tradeoff between power and delay is observed), and (ii) delay-optimal wire dimension changes according to the driver strength. The reason for (i) is that the effective resistance of the BUF_X1 is relatively larger than the resistance of the wire, which results in a larger impact of wire capacitance (compared to that of wire resistance).

Figures 4.16(a), (b), (c) and (d) show power and delay contour maps with different wirelength values, i.e., $5\mu m$, $10\mu m$, $15\mu m$ and $20\mu m$, respectively. The delay contours move toward the right and upward as the wirelength increases, as expected.

Figures 4.17(a), (b), (c) and (d) show power and delay contour maps with output load values $2fF$, $3fF$, $5fF$ and $10fF$, respectively. We observe that larger width values are preferred as the load cap increases. This might be because as the load capacitance increases, the stage delay dependence on wire capacitance lessens, and the relative sensitivity to wire resistance increases.

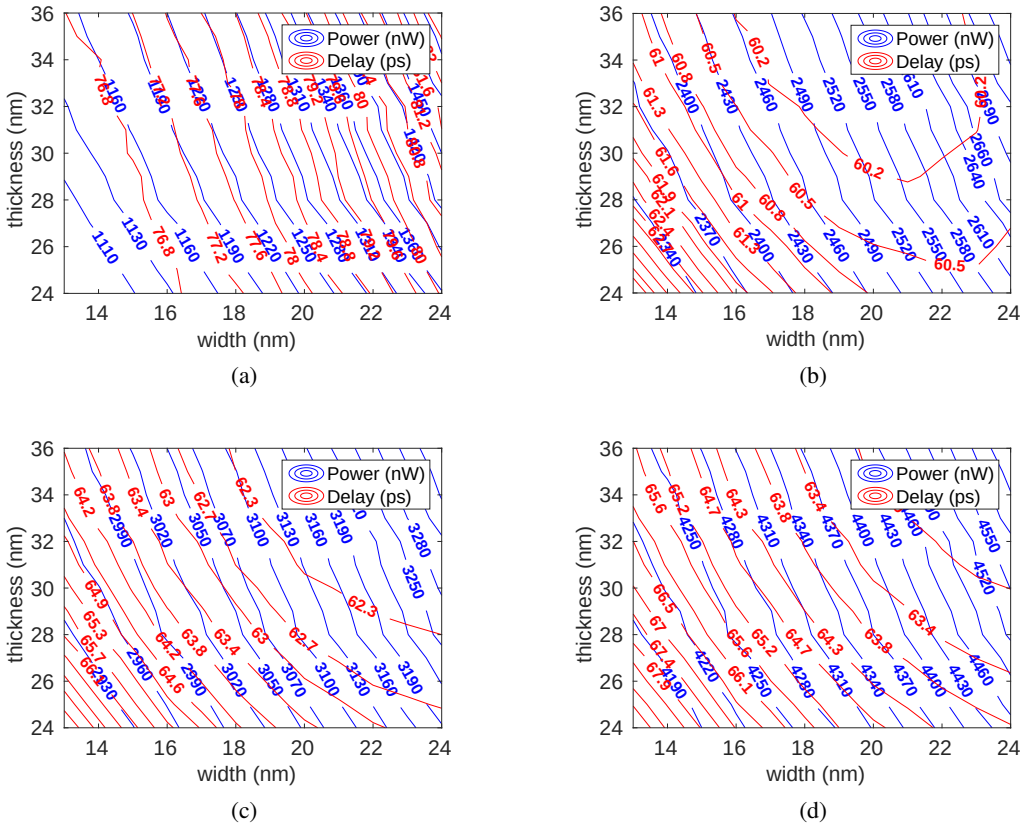
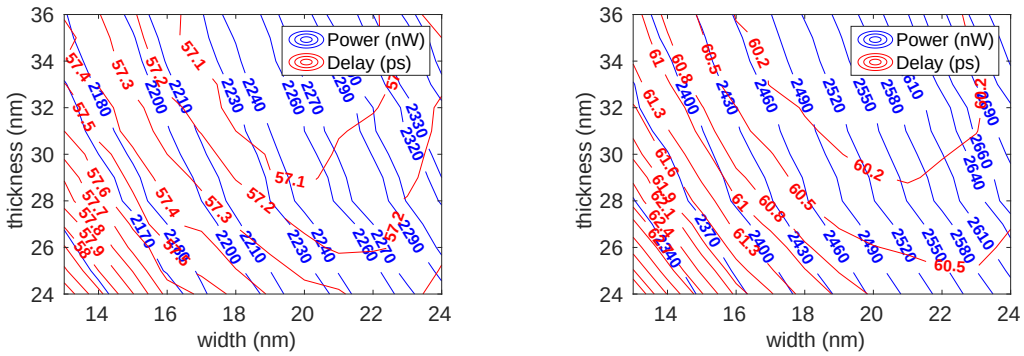
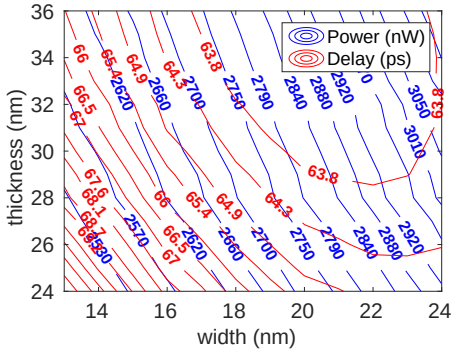


Figure 4.15: Sensitivity of power and delay to driver strength: (a) BUF_X1, (b) BUF_X2, (c) BUF_X8 and (d) BUF_X16.

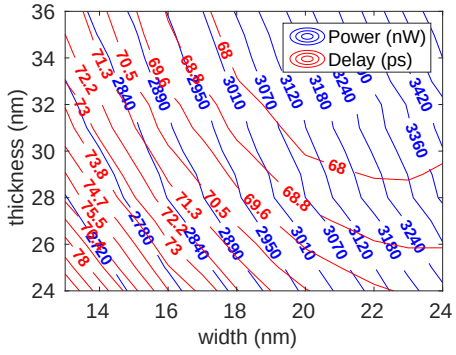


(a)

(b)



(c)



(d)

Figure 4.16: Sensitivity of power and delay to wirelength: (a) $5\mu m$, (b) $10\mu m$, (c) $15\mu m$ and (d) $20\mu m$.

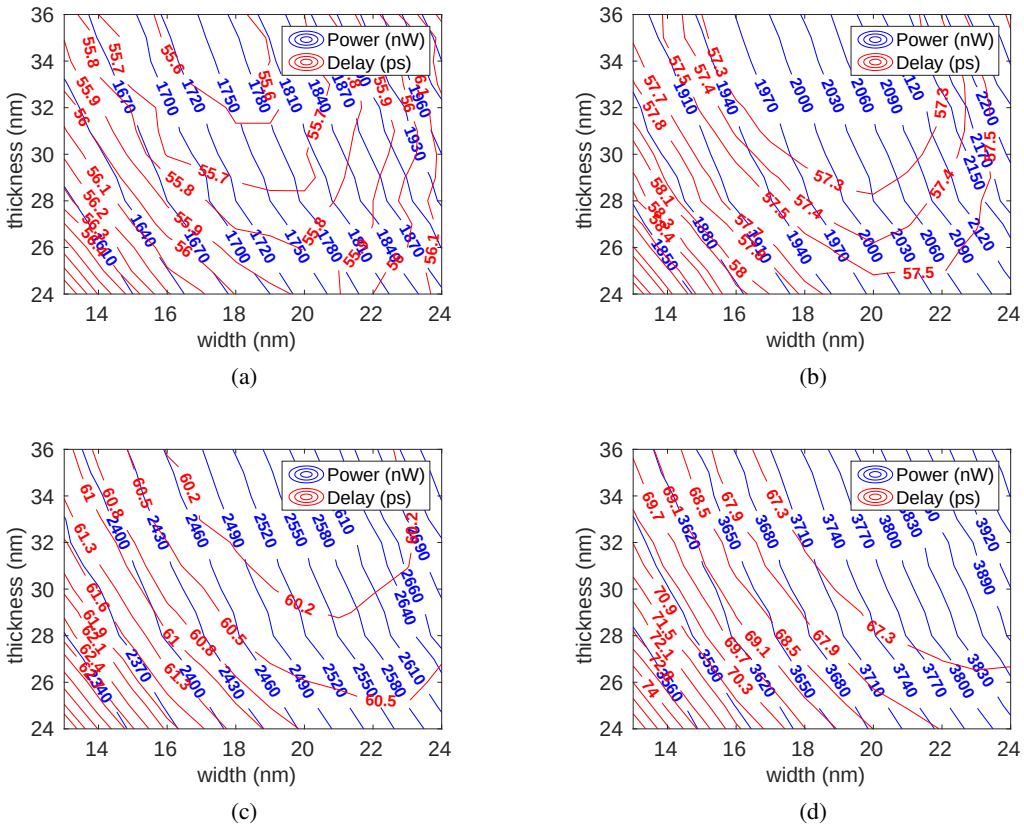


Figure 4.17: Sensitivity of power and delay to output load: (a) $2fF$, (b) $3fF$, (c) $5fF$ and (d) $10fF$.

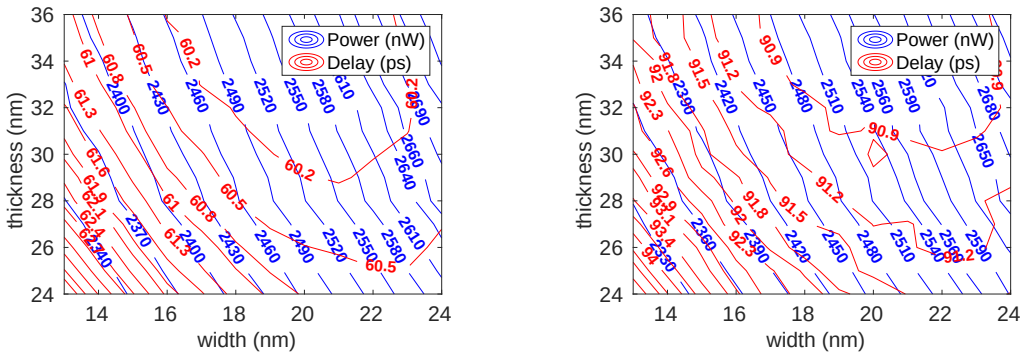


Figure 4.18: Sensitivity of power and delay to input slew: (a) 50ps and (b) 100ps.

Figures 4.18(a) and (b) give power and delay contours showing the sensitivity to input slew. Although the absolute delay and power values are different, the relative delay and power values do not change significantly, suggesting that input slew is not a critical factor in determining power- and/or delay-optimal wire dimensions.

Validation on Single-Stage Paths

We have confirmed consistency between our SPICE-based results and the block-level analysis flow within a commercial P&R tool [174]. As shown in Figure 4.19, for each type of layer, we generate eight bits[37] of signal wires, and compare (i) the stage delay of the middle signal as reported by the P&R tool's static timing analysis (STA) capability to (ii) the delay reported from SPICE simulation. To avoid effects of via parasitics, the drivers are located at each wire input port with modified driver output pins on the metal layer of the wire. For each type of layer, we sweep DC (i.e., 0.4, 0.45, 0.5, 0.55, 0.6, 0.65 and 0.7) and AR (i.e., 1.5, 1.75, 2.0). We use three wirelength values (i.e., 100μm, 200μm and 300μm) and three output load values (i.e., 5fF, 10fF and 15fF). Figure 4.20 plots the delays reported by the P&R tool and SPICE simulation, suggesting strong correlation of STA in P&R with SPICE simulation.

[37] Our background study indicates that for each signal wire, more than one neighboring signal wire contributes to its capacitance. Out of eight parallel wires, we consider the fourth and fifth to be "middle" wires.

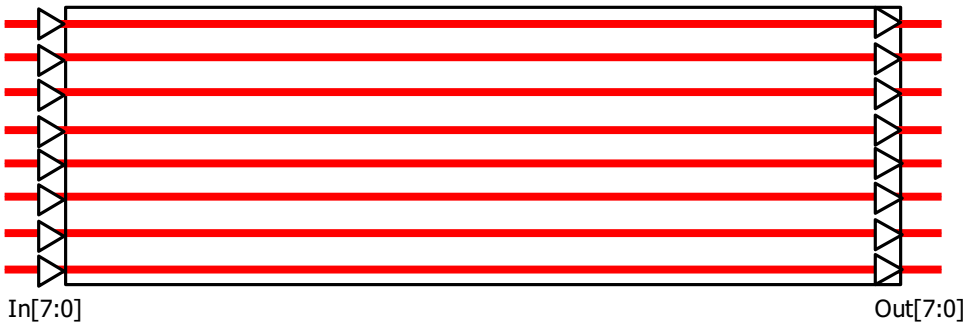


Figure 4.19: The artificial testcase with eight bits of single-stage paths.

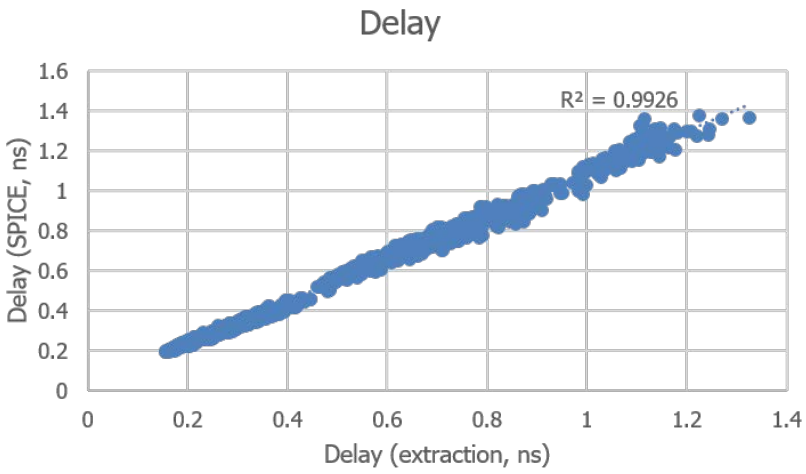


Figure 4.20: Correlation between timing reports from P&R timing analysis and SPICE simulation.

4.2.3 Block-Level Validation

For block-level validation on real designs, we enable a commercial P&R tool flow with the following steps. (i) We first group cells by their driver strength, and place-and-route using *Cadence Innovus* [174] within each group. From SPICE-based simulation studies, driver strength is the key factor in determining optimal wire dimensions. Thus, by limiting the range of driver strength of cells, we can find better correlation to SPICE-based results. In our implementation, we partition the cells into two groups, with driver strengths of $\times 1$ and $\times 4$, respectively. (ii) To avoid the tool's noise, we use one fixed post-routed layout for a design with default wire dimensions (i.e., AR[38] = 2.0, DC = 0.5) for all metal layers. Then we extract parasitics by using different BEOL stack by varying (AR,DC) combinations, and report design

[38]For consistency over all metal widths, AR is henceforth defined as metal thickness divided by metal half-pitch.

metrics (timing and power). For DC, we sweep from 0.5 to 0.7 with $1nm$ step in wire width for $1\times$ and $1.5\times$ layers, and with $2nm$ step for the $2.5\times$ layer. For AR, we use 1.50, 1.75, 2.00 and 2.25 for selection. In our experiment, we run our extraction flow for all (AR,DC) combinations for each layer type, while fixing the other layer types with the default configuration.

Figures 4.21 and 4.22 show the contour maps of delay (power) from both SPICE simulation and place-and-route runs. For SPICE simulation, we assume a wirelength of $10\mu m$ and FO3 capacitance load.[39] For block-level validation, we use a low-density parity-check (LDPC) decoder block [191] as our reference design. We report the contour maps of delay and power, with varying metal width and thickness. From Figures 4.21(a) – (c), we can see that for the $\times 1$ cell group, there is no tradeoff between power and delay. Therefore, optimal power and delay are always achieved with smaller width and thickness, which is verified in Figures 4.21(d) – (f). From Figures 4.21(a) – (c), we see that there is a tradeoff for the $\times 4$ cell group. For better delay, medium (resp. small) DC is preferred with higher AR for $1\times$ (resp. $1.5\times$) metal layers, while smaller (AR,DC) is preferred for $2.5\times$ metal layers, which is also verified from our block-level design.

For both cell groups, smaller (AR,DC) is always preferable in terms of power, due to smaller capacitance; this can be seen in Figures 4.23(d) – (f). To create a simplified real-world configuration for high-performance blocks, we rerun the P&R flow enabling both $\times 1$ and $\times 4$ cells with the tightest clock period achievable[40] and we plot the contour maps in Figure 4.23. We also show the wirelength distribution per layer type, labeled by the cell group of the driver. As shown in Table 4.4, since $\times 4$ cells drive more than double the wirelength on every layer, the contour plot is more similar to that of $\times 4$'s.

Overall, if the designs with $\times 1$ cells can be seen as low frequency and low power, and designs with $\times 4$ cells can be seen as high frequency and high performance, our observations show that those designs prefer distinct BEOL stacks, as shown in Figure 4.12. For a simplified real-world high-performance configuration with cells of multiple driver strengths, the above preference of BEOL stack from high-drive cells still holds as larger cells drive at least $2\times$ wirelength on each metal layer, suggesting that optimization of (AR,DC) towards high-drive cells may be beneficial for every layer.

[39] In our background study, we place-and-route seven designs from the *OpenCores* website [191], observing average net length of $2\mu m$ to $8\mu m$, and average fanout of approximately three for each design. A similar configuration is used in [30].

[40] A timing target is considered to have been achieved if setup worst negative slack (WNS) $>$ -50ps.

Table 4.4: Wirelength distribution per layer type (normalized) grouped by driver cells.

Layer		**1×**		**1.5×**		**2.5×**	
Driver		**×1**	**×4**	**×1**	**×4**	**×1**	**×4**
Design	AES	0.15	0.41	0.06	0.31	0.02	0.05
	LDPC	0.11	0.21	0.03	0.28	0.02	0.36

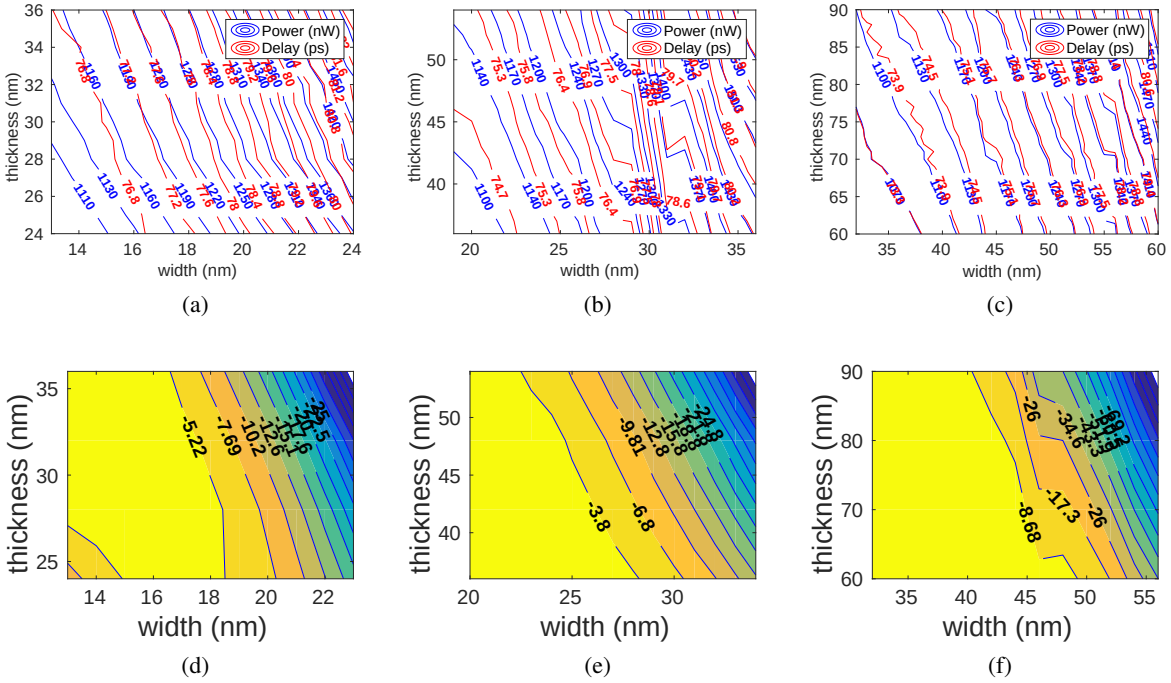


Figure 4.21: Block-level validation to single-stage SPICE simulation: (a) – (c) contour maps of power and delay (BUF_X1) for 1×, 1.5× and 2.5× metal layers, respectively, assuming wirelength of $10\mu m$ and load of $2fF$; (d) – (f) contour maps of TNS (total negative slack) when varying (AR,DC) for 1×, 1.5× and 2.5× layers, respectively.

4.2.4 Experimental Setup and Results

In this section, we explore the potential future benefits of design-aware manufacturing (DAM) and manufacturing-aware design (MAD) [54] methodologies. Design-aware manufacturing refers to the optimization of manufacturing to maximize the quality of a given product design. Here, DAM means that (AR,DC) is tuned according to the characteristics of each design. For example, in a DAM flow, we may propose a specific BEOL stack for a given P&R solution. Manufacturing-aware design refers to the optimization during physical implementation, where a downstream, post-P&R optimization of (AR,DC) is assumed in the P&R flow, and thus is applied in both P&R and manufacturing.

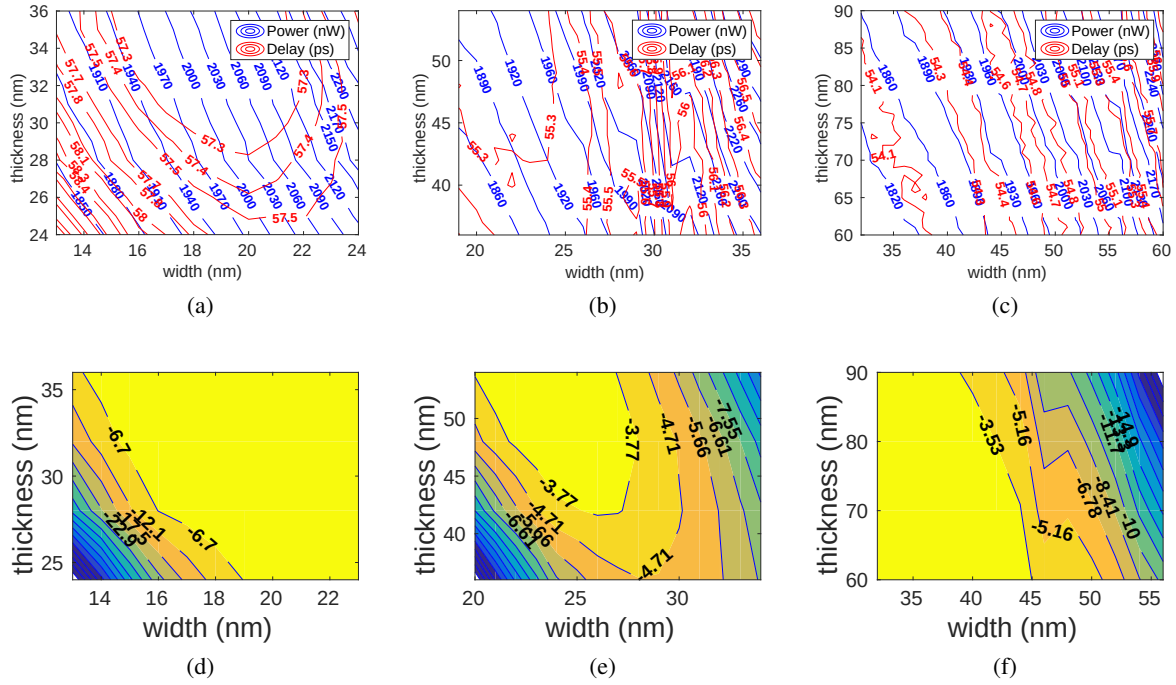


Figure 4.22: Block-level validation to single-stage SPICE simulation: (a) – (c) contour maps of power and delay (BUF_X4) for 1×, 1.5× and 2.5× metal layers, respectively, assuming wirelength of $10\mu m$ and load of $3fF$; (d) – (f) contour maps of TNS (total negative slack) when varying (AR,DC) for 1×, 1.5× and 2.5× layers, respectively.

Experimental Setup

We investigate the design freedoms of DAM / MAD, and their impacts, for a total of 108 BEOL stacks covering wide (AR,DC) ranges as follows. (i) We perform P&R using the 108 BEOL stacks. By covering a variety of BEOL stacks, we effectively explore MAD. (ii) For each implementation with a different BEOL stack, we perform PEX and STA using all BEOL stacks of the DAM study. This step shows DAM for each stack.

After P&R and PEX, we report the total negative slack (TNS) and the total power from STA for 108×108 data points (the total number of pairs of a MAD stack and a DAM stack). In the following discussion, we use the naming convention *(P{stack number}, R{stack number})* to represent the pair of a MAD stack and a DAM stack.

In our BEOL stack, we have two 1× layers, two 1.5× layers, and four 2.5× layers. For each type of layer, we choose from three DC combinations (i.e., 0.5, 0.6, 0.7). For all the layers, we apply a uniform

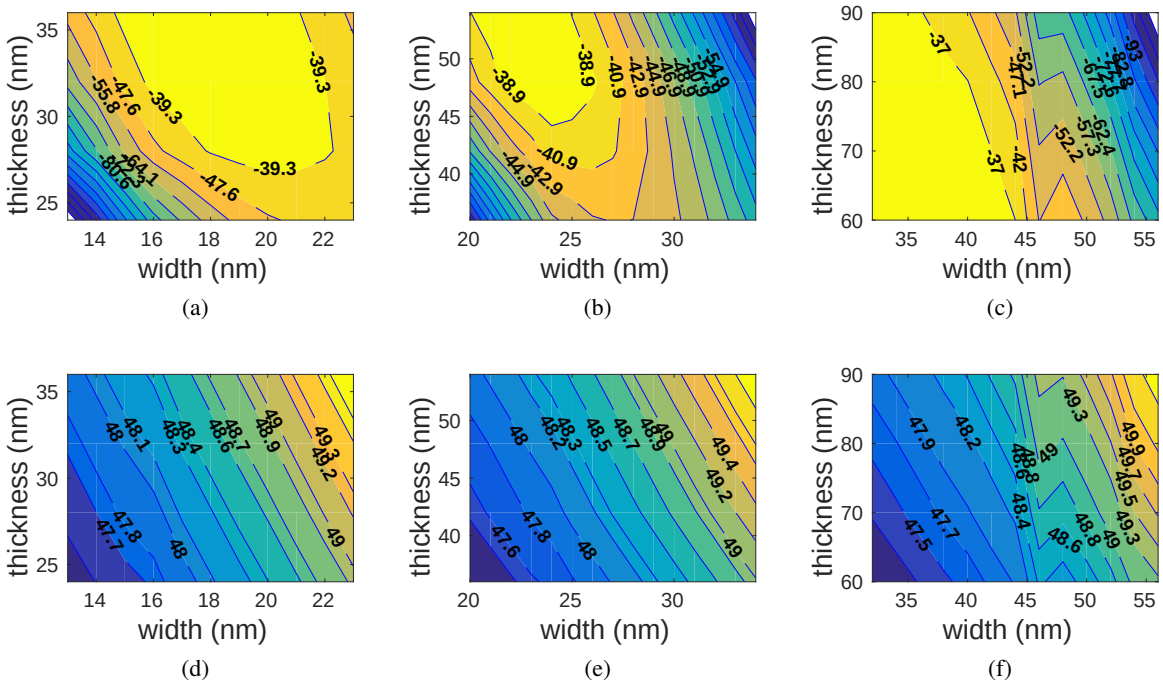


Figure 4.23: Contour maps of (a) – (c) TNS (total negative slack) and (d) – (f) power when varying (AR,DC) for 1×, 1.5× and 2.5× layers, respectively.

AR from four combinations (i.e., 1.5, 1.75, 2.0, 2.25) to reduce the number of combinations. Thus, we have a total of $3 \times 3 \times 3 \times 4 = 108$ BEOL stacks. Table 4.5 summarizes a noteworthy subset of the BEOL stack configurations.[41] All experiments are performed at the typical-typical (TT) process corner, for the LDPC testcase with clock period of $0.8ns$, and with availability of both $\times 1$ and $\times 4$ cells.

Table 4.5: A noteworthy subset of BEOL stack configurations.

Stack	1×		1.5×		2.5×	
index	**AR**	**DC**	**AR**	**DC**	**AR**	**DC**
1	1.50	0.5	1.50	0.5	1.50	0.5
4	2.25	0.5	2.25	0.5	2.25	0.5
18	1.75	0.5	1.75	0.6	1.75	0.6
25	1.50	0.5	1.50	0.7	1.50	0.5
55	2.00	0.6	2.00	0.6	2.00	0.6
108	2.25	0.7	2.25	0.7	2.25	0.7

[41] Index $= 36 \cdot i + 12 \cdot j + 4 \cdot k + l$, where i, j and k are the indices of DC for 1×, 1.5× and 2.5× layers, respectively, and l is the index for AR. See the examples in Table 4.5.

Experimental Results

Figure 4.24 shows the results of the DAM and MAD studies. In the figure, the x-axis gives 108 different BEOL stacks[42] for P&R, and the y-axis gives both TNS and total power. For each P&R stack, we give the TNS after PEX and STA with the default BEOL stack in red dots and the respective power is represented in black bars. Red columns show TNS after PEX and STA with all 108 stacks, while the orange columns show the power, also for all 108 stacks. We sort the P&R stack indices along the x-axis, according to TNS based on the default BEOL stack. In this way, the impact of DAM is presented horizontally (by comparing between red dots and black bars), and the impact of MAD is presented vertically (shown in red and orange bars), for each P&R stack.

The red dots indicate that different physical implementations may result in up to 40% difference in TNS. For example, the TNS difference between the leftmost BEOL stack (P55,R55) and the rightmost BEOL stack (P25,R25) in red dots is 2.59ns (40% of TNS with (P55,R55)). This means that TNS can be improved by up to 40% by exploiting the DAM. Regarding the MAD, we observe that for the P&R implementation with the default BEOL stack P108, the TNS can vary from -9.15ns to -5.28ns. This means that we can improve the TNS by up to 49%. By combining the DAM and MAD exploration, the maximum improvement of the TNS is from -9.15ns ((P108,R1)) to -3.66ns ((P18,R4)), which is a 60% improvement, albeit dependent on the given timing specification.

The black bars in Figure 4.24 indicate that different physical implementations may result in up to 7% difference in power. Also, we can observe that the red dots and black bars stay steady for each red and orange column, suggesting that given a routed design, a particular BEOL stack may be preferred regardless of the BEOL stack used for P&R. Even though we sort the P&R stack by TNS, we cannot find a monotonic trend for power, suggesting a weak correlation of timing and power for each design.

4.2.5 Conclusion

In this work, we have investigated the potential impact of design-aware manufacturing (DAM) and manufacturing-aware design (MAD) methodologies to optimize BEOL dimensions in sub-10nm nodes.

[42]Due to the limited space, we do not show the names of all the 108 BEOL stack options in the chart. Wire dimension information of noteworthy BEOL stack options is shown in Table 4.5.

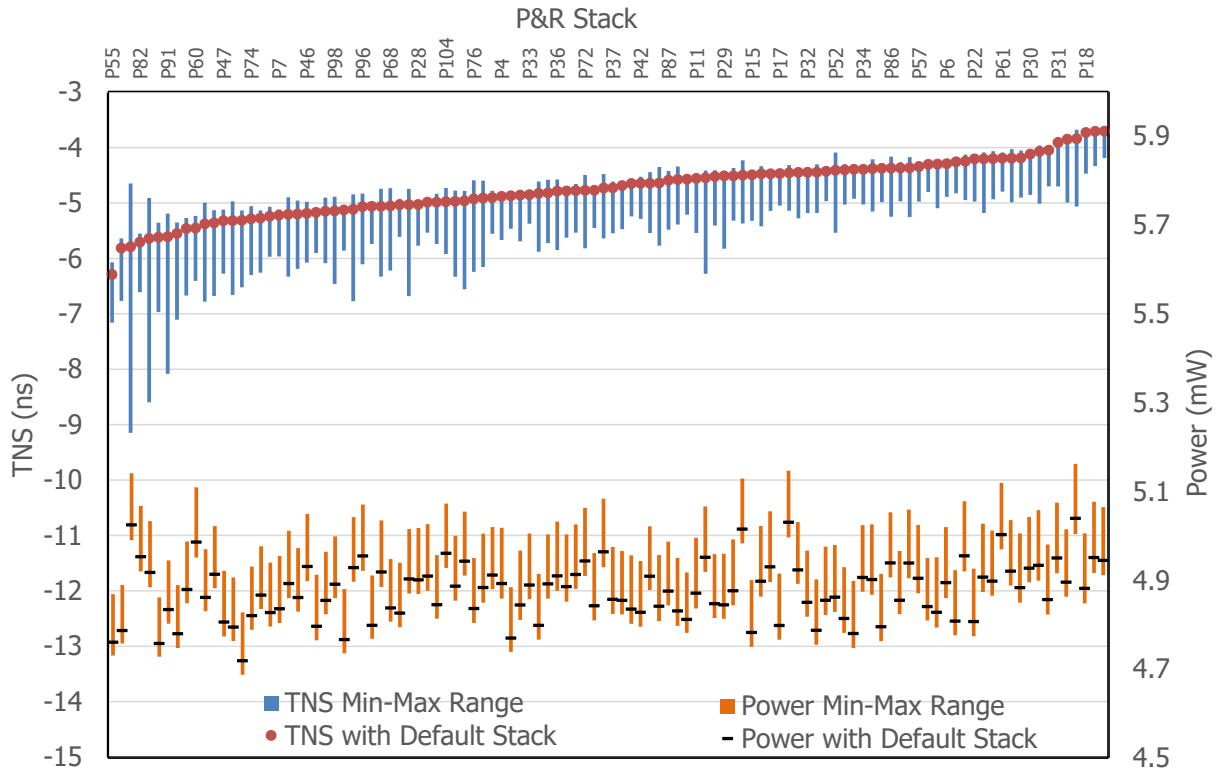


Figure 4.24: Study of DAM and MAD.

We study BEOL interconnect stack geometry by exploring the wire aspect ratio (AR) and duty cycle (DC). We perform SPICE-based analyses of timing path delays and correlate these with analyses in the P&R tool, using a single-stage artificial netlist construction. We also perform block-level studies with placed and routed designs. Based on our studies, we find the optimal (AR,DC) for a given wire pitch with respect to power and delay; we also show the sensitivities of BEOL stack geometry to circuit parameters and validate our SPICE analyses with real block-level designs. We further perform studies on design-aware manufacturing and manufacturing-aware design to explore the design freedoms and potential benefits of DAM and MAD. Large differences in design metrics exist across DAM and MAD. By proper utilization of DAM and MAD, we can save up to 60% in TNS and 7% in power for a particular LDPC testcase. Furthermore, based on our experiments, we conjecture that an optimal MAD and DAM BEOL stack exists for any given design.

Our future works include (i) the co-optimization of the front-end (i.e., gate sizing / buffer insertion, etc.) with the back-end (BEOL stack geometry); (ii) study on the impact of airgap layer and airgap-aware

BEOL stack optimization. More specifically, we hope to study "chicken-and-egg" loop for MAD and DAM, where P&R is guided by the input BEOL stack option and the netlist changes accordingly, while the optimal BEOL stack option changes according to the design (netlist) information, such as driver strength, wirelength and slack distribution. Figure 4.25 shows an example flow for co-optimization of the design implementation front-end with the manufacturing technology back-end. In this flow, we suggest a big-loop optimization that considers the interactions between P&R and optimal BEOL stack options (including airgap layers).

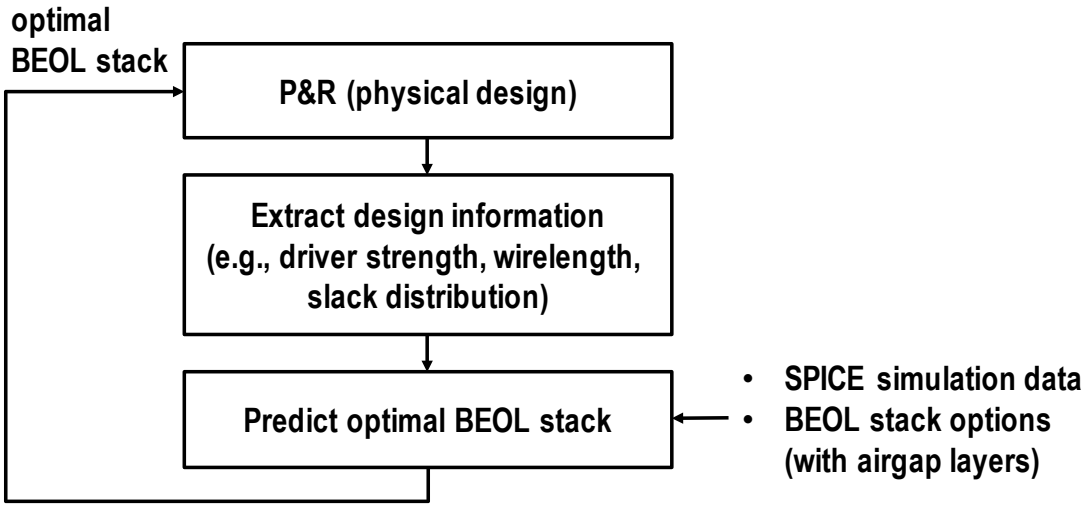


Figure 4.25: Co-optimization of SoC physical implementation (design process) with BEOL stack optimization (manufacturing process). BEOL stack options include airgap layers.

4.3 PROBE: A Placement, ROuting, Back-End-of-line Measurement Utility

Particularly in advanced technology nodes, interconnects significantly affect the power, area and performance of integrated circuits. Requirements of high integration density and performance, as well as patterning technology, design rules and cost implications, make it imperative to determine good back-end-of-line (BEOL) stack options for sub-22nm technology nodes. Yet, to our knowledge, there has been no systematic framework to measure the routing capacity of a BEOL stack for a given router, or for a combination of router and placement, particularly in a tool-agnostic, intrinsic sense. Moreover,

measurement of the routing capacity is nontrivial due to the gear ratio of metal pitches, via blockages and many other factors. A common methodology to measure the routing capacity of a given BEOL stack will simply perform routing on instances of placed designs (e.g., placement solutions for different netlists, possibly with different utilization and aspect ratio configurations). However, the routing outcomes will highly depend on the input netlist, placement solution quality and technology (e.g., standard cell architecture and track height, BEOL design rules). It has not been previously contemplated in the literature that design-technology co-optimization might be supportable with a "quasi-universal" routing capacity rank-ordering of alternative BEOL stacks, where this rank-ordering is – empirically – general across different netlists and placement solutions.[43]

In this work, we propose a general framework to measure the routing capacity of a BEOL stack with a given router. Based on the gradual perturbation of an initial placement of a netlist topology (e.g., a 2D mesh), our framework is able to obtain a ranking of alternative BEOL stacks according to routing capacities, for a given router. Quite interestingly, we further experimentally confirm that the measurement (i.e., routing capacity ranking of BEOL stacks) based on mesh-like placements is largely consistent with those based on placements generated by a commercial P&R tool within a standard SP&R flow (see Section 4.3.3 below). Moreover, our results appear, empirically, to be robust across various mesh-like placements with different characteristics (e.g., types of cells, track height, number of instances, row utilization, pin alignment and 1D/2D routing).

Our proposed framework enables new insights into important questions regarding BEOL stack options, such as "In terms of routing capacity, one P100 ($100nm$ pitch) layer is equivalent to how many P150 layers, for a given router?" and "On which layer(s) is it most valuable to enable bidirectional routing?" By gradually perturbing mesh-like placements, we also generate placements with different routing "hotspot" sizes.[44] We study the relation between routing hotspot size, placement quality (i.e., indicated by the amount of perturbation away from the optimal mesh-like placement), and the number of post-route design rule violations (#DRVs). Last, we present an analytical study based on exponentiation of a Markov transition

[43] In this work, we use the term "quasi-universal" to refer to rank-orderings of BEOL stacks that are empirically highly correlated (e.g., correlation coefficient > 0.9) across different designs and/or design enablements (e.g., different netlists, placers and routers).

[44] Following common usage in the IC and EDA fields, we use the term "hotspot" to refer to a local window of the layout that is spatially co-located with routing failure.

matrix to demonstrate how, with the same placement density and quality, a larger design is more likely to experience routing failures, an observation which is also supported by our empirical data. Our contributions are summarized as follows.

- We propose a systematic framework to assess routing capacity of BEOL stack options for a given router.
- We propose a novel metric, K, that intuitively corresponds to a systematic worsening of placement quality for a given standard-cell netlist – achieved by iteratively swapping pairs of adjacently-placed cell instances – starting from a given initial placement.
- By sweeping K, we determine K_{th} as the minimum K that results in routing failure.[45] We apply K_{th} in our routing capacity assessment.
- We empirically demonstrate a quasi-universal rank-ordering of K_{th} for BEOL stack routing capacity across placements generated by a commercial P&R tool and simple mesh-like placements.
- We show that the proposed framework can be used to evaluate routers.
- We study the size and placement quality of a routing hotspot and their correlation with #DRVs after routing.
- We perform both an analytical study and an experimental study to demonstrate how, with the same placement density and quality, a larger design is more vulnerable to routing failures.
- We suggest the possibility of a technology-dependent sweetspot of block size (trading off overheads of design decomposition against overheads of routing difficulty) that provides the best QoR in SoC implementation.
- A utility that implements the above framework for arbitrary enablement (LEF, .lib) is available at [201].

[45] In this work, if a routed design has #DRVs > 150, we consider it as a design with routing failure.

The mesh-like placements in our framework cannot perfectly represent real-life placements which comprehend timing, signal integrity, power integrity and other conflicting objectives and constraints that circuit designers face. However, rather than generating testcases that mimic real designs, our approach enables systematic generation of benchmarks with gradually increasing routing difficulty. This enables a principled rank-ordering of the respective routing capacities of different BEOL stacks, which to our knowledge cannot be straightforwardly achieved using specific real design instances. Our work empirically shows that dependence on specific design instances may be avoidable, which would help resolve a long-standing controversy regarding the utility of artificial versus real testcases. In sum, our results suggest that the mesh-like placements may be usable as proxies of real-life placements, at least to rank-order BEOL stack options in terms of routability. The routability-oriented rank-ordering can help reduce the number of configurations that must be considered as product teams address the complex BEOL stack optimization problem.

4.3.1 Related Work

In this section, we review the previous literature on (i) benchmark construction and (ii) routability estimation.

Benchmark Studies

Many works have evaluated VLSI optimizations and design enablements using various kinds of benchmarks. This literature can be divided into two basic categories: (i) artificial benchmark generation, and (ii) realistic benchmarks.

Artificial benchmarks. To evaluate the performance of VLSI optimizations, a number of artificial benchmark generation approaches have been proposed in previous literature. These include *circ/gen* [71], *gnl* [145] and the work of [36]. In the interests of realism, *circ/gen* [71] measures characteristics (e.g., circuit size, number of IOs, path depth and fanout distribution) of existing circuits and uses these characteristics as constraints in its synthetic circuit generation. As an extension to graph-based benchmark generation (which only considers graph-based properties such as Rent parameter [96] and net degree distribution), *gnl* [145] considers functional information, in that it uses a specified component library

and avoids combinational loops. The authors of [36] propose a method for generating random circuits for routability measurement. The input parameters of their benchmark generation include the size of the design, the Rent parameter and the number of IOs.

Several methodologies produce instances with *known optimal solutions*, which enables quantification of a heuristic optimization's suboptimality. For placement optimization, the **PEKO** benchmark generator [25] provides a netlist (with user-specified number of placeable modules and net degree distribution) as well as a constructive placement solution with known minimum wirelength. Attributed to Boese in [59], **PEKU** [31] further improves the realism by including non-local nets, while generating instances with known upper bounds on optimal wirelength. In a similar spirit, there are also gate-sizing-oriented benchmarks with known optimal solutions. The work of [55] generates benchmark circuits (called *eyecharts*) of arbitrary size along with a method to compute their optimal solutions using dynamic programming. The work of [83] generates more realistic benchmarks (by comprehending path depth and fanin / fanout distributions) with known optimal solutions for gate sizing problems.

Realistic benchmarks. Artificial benchmarks with known optimal solutions can help quantify suboptimality of heuristics for NP-hard optimizations. However, their artificial nature has lessened their impact among practitioners. Certain methodologies quantify suboptimality of heuristics for hard VLSI optimizations based on transformations of existing realistic benchmarks. Hagen et al. [59] propose a general measure of heuristic performance based on the notion of *scaling suboptimality*. The authors construct scaled VLSI instances from initial VLSI instances (e.g., by replicating a netlist or connecting together multiple copies of a netlist) and use these to obtain quantified lower bounds on the suboptimality of VLSI layout heuristics such as placers and partitioners. Kahng and Reda [90] propose *zero-change transformations* to quantify the suboptimality of existing placers. Given a netlist and its placement from a placer, their zero-change transformations alter the given netlist while keeping its half-perimeter wire length (HPWL) constant, resulting in zero change to achievable HPWL. In [90], the authors show that placers fail to attain their original HPWL results, with large deviations, on the altered netlists. Their work can provide suboptimality information with respect to a given arbitrary (real) benchmark.

However, none of these previous benchmarks and benchmark generation methodologies helps to evaluate *technology itself*, or the capability of tools in a technology-dependent manner. For example,

it would be enormously valuable if semiconductor product companies, foundries and equipment makers could measure the related routing capacities of alternative BEOL stacks for given combinations of, e.g., placement and routing tools. In this work, we propose benchmarks and methodologies to measure the routing capacity of BEOL stacks for a given router.

Routability Studies

Literature on routability estimation can be divided into the categories of: (i) BEOL stack options, and (ii) estimation of congestion.

BEOL stack options. Only a few works in previous literature study the design enablement (i.e., BEOL stack options) in terms of routability. Dong et al. [38] propose an analytical model to estimate the required number of metal layers for a design, based on assumed wirelength distribution and metal layer utilization efficiency. However, realistic constraints such as timing and design rules are not considered in their model. Also, the model simply assumes maximum metal layer utilization at lower layers, and does not comprehend the router's behavior. A recent work [24] develops machine learning-based models to predict whether a placement solution is routable for a given number of metal layers; based on this, an early-stage estimation of the minimum required number of metal layers can be obtained.

Estimation of congestion. Many works perform early estimation of routability and congestion of a placement or even a netlist. The works [62][94][109][124] simply use academic global routers (e.g., BFG-R [68] or FastRoute 2.0 [125]) to estimate routing congestion. [15] uses pin density and constructs Steiner trees (where multi-pin nets are split into many two-pin nets) to estimate congestion. Liu et al. [108] estimate wirelength and routing congestion of a netlist based on net range (i.e., circuit depth spanned by all terminals of a net) and structural pin density (i.e., the ratio between total number of pins of a net versus the total pin count in the design). [142] models the routing demand by assuming a rectangular uniform wire density per net. Yang et al. [163] use Rent exponent to estimate wirelength distribution of a region, based on which they predict congestion. Taghavi et al. [148] propose a local congestion metric that indicates the routing difficulty for each cell in the design library. Wei et al. [155] estimate both global and local routing congestions. Kahng and Xu [93] comprehend blockage effects in their congestion estimation model. The congestion estimation in [110] comprehends layer directive and scenic constraints to limit the routing layer

usage and the maximum wirelength of timing-critical nets. Westra et al. [156] study the actual behavior of a routing engine. Their results show that the number of nets with detour (e.g., nets with many bends) is negligible. They also observe that the ratio between L-shapes and Z-shapes of two-pin nets is roughly a constant value. Based on these observations, they propose a fast congestion prediction model. [138] also assumes no wires are detoured and proposes an analytical model based on probabilities of v-bend paths to estimate congestion. Based on the global routing information, Zhou et al. [166] propose a learning-based model to estimate routing congestion after detailed routing. Based on pin density and congestion map from global routing, Qi et al. [126] apply multivariate adaptive regression splines (MARS) to predict routing congestion.

4.3.2 Assessment of Design Enablements

In this section, we describe our framework to measure routing capacity of BEOL stack options as well as inherent capability of routers and, potentially, placers. We summarize the notations used in this work in Table 4.6. The basic idea of our framework is as follows. We start with (an instance of) a placed netlist that is straightforward to route, i.e., the routing can be completed by the routing tool with no design rule violations. We then gradually perturb the placement by randomly swapping the placed locations of adjacently-placed cell instances (i.e., a neighbor-swap operation) to increase the routing difficulty.[46] After a number of neighbor-swaps, the routing becomes infeasible (i.e., the number of post-route design rule violations exceeds a predefined threshold). We use the number of neighbor-swaps that leads to routing failures as an indicator of the routing capacity of the given BEOL stack, as well as of the capability of the router. For example, we may obtain a ranking of different BEOL stacks in terms of their routing capacities, based on the corresponding values of this indicator.

Our use of the neighbor-swap operator is intuitively reasonable for placement perturbation, for at least two reasons. First, as observed by Alpert et al. [7], with high utilizations the placement problem reduces to the ordering problem since there is not much whitespace in which cells can move. Second, a neighbor-swap is an intuitive quantum of suboptimality or error in placement; in this light, starting from a

[46] We use a discrete uniform distribution to randomly select a cell, and then a random neighbor of that cell, for swapping. Thus, it is possible for one random swap to revert a previous random swap.

mesh (Rent p = 0.5) that is perfectly placed, and then executing random neighbor-swaps, intuitively allows us to dial up particular amounts of suboptimality in placement.

The key benefit of our approach is its ability to systematically evolve a placement from routable to unroutable; this enables evaluation and ranking of BEOL stacks in terms of their routing capacities. It is more challenging to perform such a ranking with real designs, due to larger instance counts (making high-volume experimentation more time-consuming), non-uniformity of topology, and non-uniformity of cell sizes. Further, with real designs, the transition from routable to unroutable is often much more abrupt than with the mesh-based netlists and initial placements that we use. This being said, Section 4.3.3 below confirms the robustness of our analyses and conclusions in multiple ways, including with real netlists and non-uniform (real) cell sizes.

Table 4.6: Description of notations used in our work.

Term	Meaning
D_h	gate-level netlist (index of netlist $\equiv h$)
P_k	placer (index of placer $\equiv k$)
$\Omega_{h,k}$	placement solution of netlist D_h using placer P_k
U	placement utilization
R_j	router (index of router $\equiv j$)
B_i	BEOL stack (index of BEOL stack $\equiv i$)
N	square root of the total number of instances in a design
J	number of neighbor-swaps
K	number of neighbor-swaps normalized to the total number of instances N^2 ($= J/N^2$)
K_{th}	minimum K value that results in routing failure
$\Pi_B^{R_j}(D_h, P_k)$	BEOL ranking in terms of routing capacity on $\Omega_{h,k}$ using R_j
ED	sum of edge distances normalized to the total number of instances N^2 (See Section 4.3.4)
CC	crossing count normalized to the total number of instances N^2 (See Section 4.3.4)
ED_{th}	minimum ED value that results in routing failure
CC_{th}	minimum CC value that results in routing failure

Our Goal

Given a netlist D_h, we place it using the placer P_k. We then apply our proposed methodology (described below) to measure routing capacities of BEOL stacks $\{B_1, B_2, ..., B_i\}$ for a given router

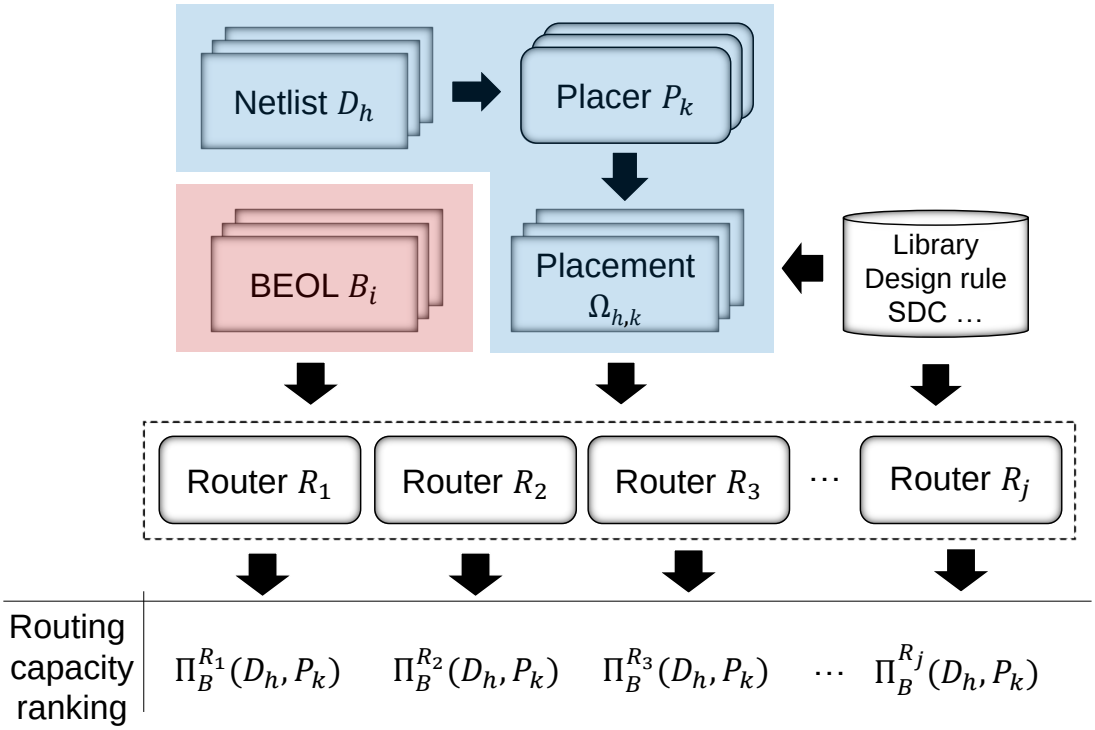


Figure 4.26: Our overall goal: determine whether it is possible to find a quasi-universal ranking of BEOL stacks in terms of routing capacity, and potentially a ranking of place-and-route tools as well.

R_j. We denote the ranking of BEOL stacks in terms of routing capacity as $\Pi_B^{R_j}(D_h, P_k)$. Our goal is to determine a quasi-universal ranking of BEOL stacks (in terms of routing capacity) for a given router, that is, $\Pi_B^{R_j}(D_h, P_k) = \Pi_B^{R_j}(D_{h'}, P_{k'})\ \forall h,\ k,\ h', k'$. In this work, we measure the routing capacity of 73 BEOL stack options with various D_h, P_k and R_j. Detailed information of the 73 BEOL stack options is given in Section 4.3.3.

Mesh-like Placement

We first study routing capacity of BEOL stacks based on mesh-like placements. We create a netlist having a square mesh topology (with M_r rows indexed by p and M_c columns indexed by q) using a given 2-input or 3-input cell. In the netlist, we connect the output pin of the gate instance with index (p, q) to input pins of the gate instances with indices ($p + 1$, q), (p, $q + 1$) and ($p + 1$, $q + 1$).[47] We then place the netlist (according to its mesh topology) uniformly in a $W_{die} \times H_{die}$ region, where $H_{die} = M_r \cdot H_{gate}$ and

[47]For a netlist composed of 2-input cells, we only connect the output of (p, q) to the inputs of ($p + 1$, q) and (p, $q + 1$).

$W_{die} = M_c \cdot W_{gate}/U$. Here, W_{gate} and H_{gate} are respectively the width and height of the given cell, and U is the predefined placement (row) utilization. We note that all gate instances have the same size.

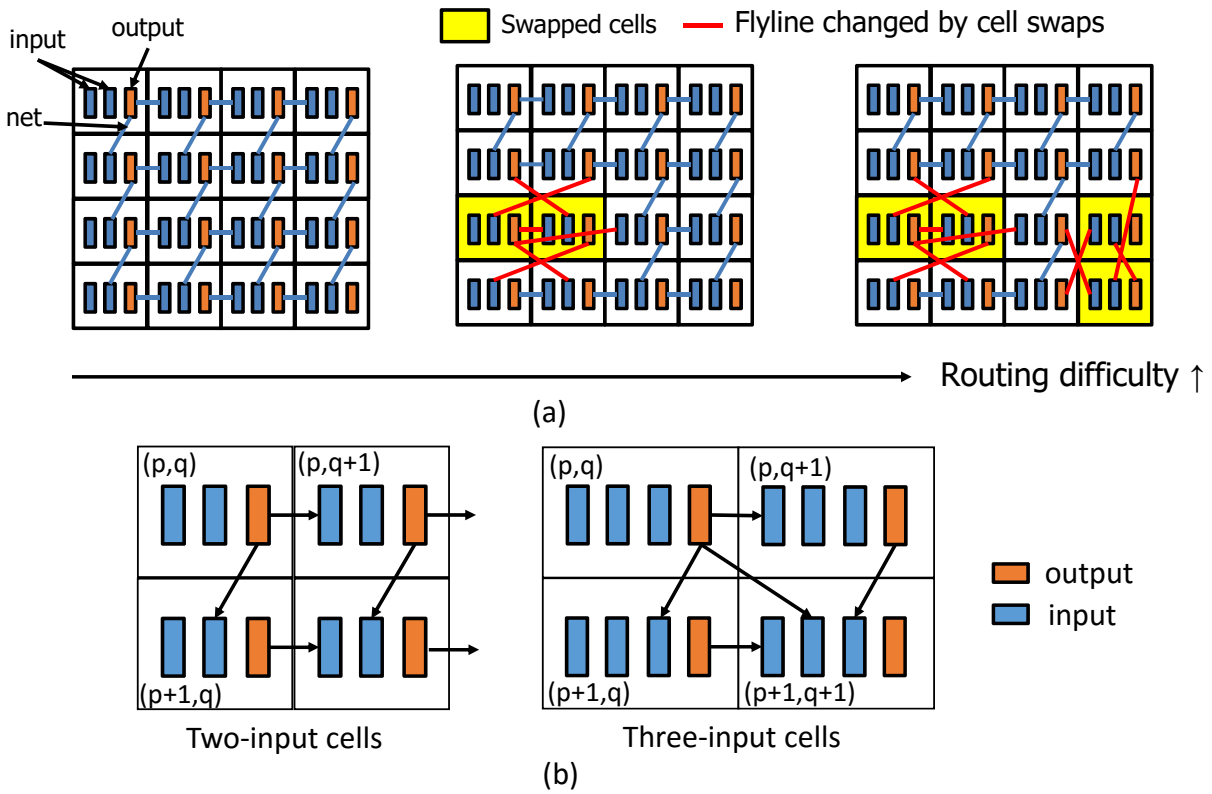


Figure 4.27: (a) Illustration of mesh-like placement and perturbations by two neighbor-swap moves. (b) Connections for 2-pin cells and 3-pin cells.

The initial mesh-like placement, where all gate instances are connected only to their physically adjacent gate instances, is easy to route. We gradually increase the routing difficulty by iteratively swapping adjacently-placed gate instances. In each move, we randomly select a gate instance and then swap it with one of its (up to four) adjacently-placed gate instances (up, down, left, right) to swap. Figure 4.27(a) shows an example with two neighbor-swap moves. Such swap moves will cause routing congestion by progressively worsening the quality (as measured by the sum of embedded edge lengths) of the placement, and eventually lead to a placement that is infeasible to route. We denote the minimum K (number of neighbor-swaps normalized to the total instance count) that leads to an unroutable placement as the K *threshold* (K_{th}). The value of K_{th} is an indicator of the routing capacity of the given BEOL stack in the

context of a (given) router, where a BEOL stack with larger K_{th} has higher routing capacity.[48] In our experiments, we perform multiple trials of the iterative swapping process for a given initial placement and report the average observed K_{th} as a measure of routing capacity. Since the K_{th} measurement requires a number of routing trials, we reduce runtime by first performing a coarse-grained search to narrow down the search space, and then performing a fine-grained search within the reduced search space.[49] It is possible to perform binary search or other search techniques to further reduce the runtime.

The mesh-like placement can be implemented with various different configurations, such as number of total instances, standard cell types, row utilizations, pin alignments, etc. To support the robustness of outcomes and conclusions, we have performed our basic experiment with various mesh-like placements with different configurations. Details are given in Section 4.3.3.

Cell Width-Regularized Placement

We also measure routing capacity based on placements generated by a commercial P&R tool, using *cell width-regularized netlists*. Our primary experiment uses bloated standard cells to help maintain placement legality as neighbor-swaps are performed.[50] Of course, a width-regularized netlist is generally not produced in the course of a usual SP&R flow. To generate a width-regularized netlist, we modify the cell LEF [186] such that all gate instances in the netlist have the same size. Here, we simply increase the width of cells without changing the layouts within the cells. We then use a commercial placer to place the netlist with bloated cells. Similar to the method for generating mesh-like placements, we iteratively swap locations of random pairs of adjacently-placed cells until the placement becomes unroutable. Again,

[48] The difference between K and K_{th} is that K is an indicator that represents suboptimality of a placement, while K_{th} is the minimum K value that results in routing failure. Thus, K can serve as a metric to evaluate the quality of a placement itself in terms of routability (i.e., K indicates the routing difficulty increase with respect to a mesh-like placement); and K_{th} can be used as a metric to evaluate the routing capacity of a BEOL stack option, or the routing capabilities of a router for a given BEOL stack option and initial placement. This is because K_{th} shows how much a BEOL stack option (or a router) can sustain (i.e., while still being able to support successful routing of the design) in terms of suboptimality of the given placement (K).

[49] In our experiments with 5K-instance designs, the runtime for routing is $\sim$10 minutes (single-threaded) on average. For most cases, the runtime for K_{th} measurement is less than 3 hours with a single core, which is a small cost for crucial technology insight.

[50] Here, we use cell bloating to make cell widths uniform (i.e., regularized), thus enabling neighbor-swaps without legalization. (When instances have different widths, placement legalization is required to ensure a legal placement after a sequence of neighbor-swaps; hence, the perturbation of the initial placement is not as gradual.) We note that to evaluate routing in terms of pin accessibility, we can use smaller, but still regularized, cell widths for all instances. Supplementary experiments use the AES and VGA designs and non-uniform (non-bloated, with original widths from the production library) cells in our framework, where we keep track of row utilization after every neighbor-swap to help maintain placement legality and to ensure the same density for all placement rows. Details are given in Section 4.3.3, below.

we find K leading to an unroutable placement (K_{th}) and use K_{th} to indicate the routing capacity of the BEOL stack. In this flow, cell bloating avoids placement legalization after each neighbor-swap move. Here the definition of adjacently-placed gate instances within the same row is trivial. However, unlike the placement of the square mesh topology, gate instances are not necessarily aligned vertically. In our experiments, for a given gate instance, we define its neighbors in the two (or, one) adjacent rows as the gate instances with the minimum center-to-center distances in horizontal direction (see Figure 4.28, left). In Figure 4.28, brown cells that have line segments drawn to each blue cell are considered as the neighbor cells of that blue cell. As with the experimental flow for **mesh-like placements**, we perform multiple runs of the iterative swapping for a given initial placement and report the average K_{th} in our experiments. Supplementary studies with two *OpenCores* [191] designs and non-uniform (non-bloated, i.e., with original, non-regularized widths) cells (see Figure 4.28, right) are also reported in Section 4.3.3.

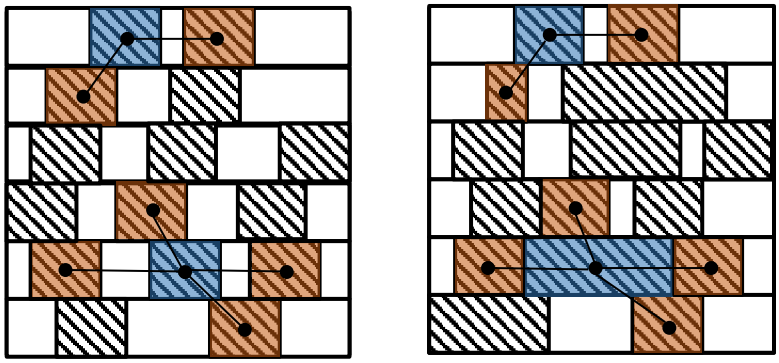


Figure 4.28: Illustration of the *neighbor* cell relation in a placement generated by a commercial P&R tool, for the cases of one adjacent row and two adjacent rows. Left: cells with uniform (bloated) widths. Right: cells with non-uniform (non-bloated) widths.

Extension to Evaluations of Placers and Routers

Similar to evaluating the routing capacity of a BEOL stack option based on K_{th}, we can also use the metric K_{th} to evaluate capabilities of placers and routers with respect to routability for a given BEOL stack option. For example, a placement that results in a higher K_{th} value for a given BEOL stack and router is more likely to be routing-friendly. Also, with respect to a given BEOL stack option and a placement, the router that achieves a higher K_{th} value is more likely to have better performance in terms

of routability.

4.3.3 Experimental Setup and Results

In this section, we describe the setup and results of three basic types of experiments.

- **Expt1:** Measurement of routing capacity of various BEOL stack options for mesh-like placements and cell width-regularized placements with two commercial routers.
- **Expt2:** Comparison of different BEOL stack options that have the same routing resources to derive new understanding of equivalences (e.g., between different-pitch layers) that can guide the choice of BEOL stack option.
- **Expt3:** Further verification of the robustness of the rank-ordering of BEOL stack options.

Experimental Setup

Testcases. In our experiments, we use mesh-like placements, and the AES encryption core and the enhanced VGA core from the *OpenCores* website [191]. Information about testcases is summarized in Table 4.7. For AES, the first utilization number is used for cell width-regularized placement; the second is used for non-bloated-cell-based placement. We use 8-track (8T) standard cells from a $28nm$ LP foundry library.[51] Mesh-like placements (e.g., *AOI-mesh*, or a mesh with AOI cells) are implemented as described in Section 4.3.2. Since routing hotspots occur locally, intuitively we do not need to use very large designs to measure the routing capacity of a BEOL stack.[52] Thus, we use 5K-instance designs. Indeed, we experimentally confirm below that the number of instances (i.e., design size) does not change the rank-ordering of BEOL stacks. Also, using small designs help to reduce the runtime. In our experiments, the runtime for routing on the 5K-instance design is $\sim$10 minutes (single-threaded) on average. The AES and VGA designs are synthesized with *Synopsys Design Compiler K-2015.06-SP4* [195] and implemented with

[51] Below, we show that results obtained using 12-track (12T) cells are consistent with those obtained using 8T cells.

[52] Routing difficulty does increase as design size increases, as quantitatively analyzed in Section 4.3.4 and as empirically demonstrated in Figure 4.35(d) below. Increased routing difficulty in larger designs reflects a higher probability of (local) routing hotspots, for a given placement quality. Indeed, the small designs used in our study can be viewed as sampled routing hotspots within larger design; see Section 4.3.4 below. From our studies, we currently believe that standard-cell netlist complexity of $\sim$15K instances (e.g., AES) or greater can afford useful conclusions regarding relative capacities of alternative BEOL stacks.

bloated combinational cells (e.g., AOI21, AOI22, BUF, INV, NAND2, NOR2, OAI12, OAI211, OAI22) and one flip-flop cell. For 8T cells, the post-bloating width of combinational cells is eight placement sites, and the flip-flop cell width, which we leave unchanged, is 23 placement sites. The average and standard deviation of width increase are 2.22 and 1.67 placement sites, respectively, for combinational cell masters.

Table 4.7: Testcases.

Name	#Inst.	#Nets	Util.	Clock period (ns)
AOI-mesh	5000	15000	90%	NA
AES	15K	15K	65% / 50%	1.4
VGA	80K	80K	45%	1.2

BEOL stack options. In the $28nm$ LP foundry library that we use, the minimum metal width and the minimum metal pitch (i.e., width + spacing) are $0.05\mu m$ and $0.1\mu m$, respectively. We introduce $1\times$, $1.5\times$[53] and $2\times$ metal layers based on these minimum width and pitch values. The width (resp. pitch) values for $1.5\times$ and $2\times$ metal layers are $0.074\mu m$ ($0.15\mu m$) and $0.1\mu m$ ($0.2\mu m$), respectively. We generate 73 BEOL stack options according to the following rules.

- The pitch and width values of M1 and M2 are not changed. The M1 and M2 pitch values are $0.136\mu m$ and $0.1\mu m$, respectively, in the $28nm$ library that we use.
- The total numbers of metal layers are five, six, seven and eight.
- Different numbers of $1\times$, $1.5\times$ and $2\times$ layers are tried.
- From M2 upward, the pitch of a given metal layer is always greater than or equal to the pitches of all lower metal layers. In other words, pitch values increase monotonically from lower to upper metal layers.[54] Also, a BEOL stack option is determined according to its numbers of $1\times$, $1.5\times$ and $2\times$ layers. For example, if #$1\times$, #$1.5\times$ and #$2\times$ layers are respectively 4, 1 and 1, then the pitch values

[53] We derive $1.5\times$ metal and via layers from the existing $1\times$ and $2\times$ layers in $28nm$ LEF since there is no $1.5\times$ layer in the $28nm$ BEOL LEF that we use. The $1.5\times$ metal width and spacing are $0.074\mu m$ and $0.076\mu m$, respectively. We use the same EOL extension spacing as seen for the $1\times$ layer; for the minimum length rule, we use the mean of the $1\times$ and $2\times$ layers' values. The minimum enclosure area rule is set to that of a $2\times$ layer. We set via spacing such that two vias cannot be placed immediately (i.e., horizontally or vertically) next to each other, but can be placed diagonally adjacent.

[54] The M1 layer is an exception to this monotonicity, since it is used for pins or internal routing within standard cells, and its pitch follows the contacted poly pitch.

of M1, M2, M3, M4, M5 and M6 are respectively $0.136\mu m$, $0.1\mu m$, $0.1\mu m$, $0.1\mu m$, $0.15\mu m$ and $0.2\mu m$.

Table 4.8 summarizes the numbers of $1\times$, $1.5\times$ and $2\times$ layers, and the *routing resources*, of all the BEOL stacks that we study, with total numbers of metal layers equal to five, six, seven and eight. There are 27 possible combinations of BEOL stack options for eight metal layers. For the cases where the total number of metal layers is less than eight, we ignore BEOL stack options with layer number larger than the specific total number of metal layers (in Table 4.8, such BEOL stack options are marked with NA).

The *routing resource* (i.e., sum of track densities per unit channel height) of each combination of (BEOL stack option, total number of layers) is calculated as $\sum_b \frac{1}{pitch_b}$, where b is a metal layer, and $pitch_b$ is the pitch value of b.[55] Note that we assume routing is unidirectional in all of our experiments, except for the experiments where we specifically study the impact of bidirectional routing (Section 4.3.3 below).

Table 4.8: BEOL stack options.

Name	#1× layers	#1.5× layers	Total number of layers = 8		Total number of layers = 7		Total number of layers = 6		Total number of layers = 5	
			#2× layers	Resource	#2× layers	Resource	#2× layers	Resource	#2× layers	Resource
BEOL0	2	0	6	40	5	35	4	30	3	25
BEOL1	3	0	5	45	4	40	3	35	2	30
BEOL2	4	0	4	50	3	45	2	40	1	35
BEOL3	5	0	3	55	2	50	1	45	0	40
BEOL4	6	0	2	60	1	55	0	50	NA	NA
BEOL5	7	0	1	65	0	60	NA	NA	NA	NA
BEOL6	8	0	0	70	NA	NA	NA	NA	NA	NA
BEOL7	2	1	5	42	4	37	3	32	2	27
BEOL8	3	1	4	47	3	42	2	37	1	32
BEOL9	4	1	3	52	2	47	1	42	0	37
BEOL10	5	1	2	57	1	52	0	47	NA	NA
BEOL11	6	1	1	62	0	57	NA	NA	NA	NA
BEOL12	2	2	4	43	3	38	2	33	1	28
BEOL13	3	2	3	48	2	43	1	38	0	33
BEOL14	4	2	2	53	1	48	0	43	NA	NA
BEOL15	5	2	1	58	0	53	NA	NA	NA	NA
BEOL16	6	2	0	63	NA	NA	NA	NA	NA	NA
BEOL17	2	3	3	45	2	40	1	35	0	30
BEOL18	3	3	2	50	1	45	0	40	NA	NA
BEOL19	4	3	1	55	0	50	NA	NA	NA	NA
BEOL20	5	3	0	60	NA	NA	NA	NA	NA	NA
BEOL21	2	4	2	47	1	42	0	37	NA	NA
BEOL22	3	4	1	52	0	47	NA	NA	NA	NA
BEOL23	4	4	0	57	NA	NA	NA	NA	NA	NA
BEOL24	2	5	1	48	0	43	NA	NA	NA	NA
BEOL25	3	5	0	53	NA	NA	NA	NA	NA	NA
BEOL26	2	6	0	50	NA	NA	NA	NA	NA	NA

[55] We use the sum-of-track-densities measure to achieve a design-independent, normalized measure for comparison of BEOL stack options. Thus, our definition of *routing resource* is a normalized routing resource. This measure reflects current usage in industry [141]. We observe that using total track length or total number of routing tracks as a measure of routing capacity would require knowledge of layout dimensions, thus introducing a design-dependence.

Expt1: Routing Capacity of Various BEOL Stack Options

In this experiment, we show our measurement of routing capacity for various BEOL stack options. We first demonstrate positive correlation between the number of DRVs (#DRVs) and K, and show how we characterize K_{th}. As detailed in Section 4.3.2 above, in our experiments we use K_{th} as an indicator of the routing capacity of a BEOL stack option (or, the routing capability of a router). We then show K_{th} versus routing resource (number of available routing tracks) of various BEOL stack options. Our experimental results suggest that routing resource may not be the only factor to determine routing capacity of a BEOL stack option. In light of this, we further study the ranking of BEOL stack options with the same routing resource based on our framework using mesh-like placement and cell width-regularized placement, with two commercial routers.

Characterization of K_{th}. We measure #DRVs to characterize the K_{th}. Figure 4.29 shows #DRVs versus K with the AOI-mesh design. For each K value, we run three trials with different random sequences of neighbor-swaps to avoid noise from tools and randomness of the perturbation process.[56] Each dot corresponds to a pair of a perturbation and a BEOL option. The average values of the sets of three runs are marked by the solid traces. We increase K until the average #DRVs > 150. We then record the current K as K_{th}. In Figure 4.29, K_{th} values are zero, four, 11, and $>$14 for BEOL0_6, BEOL1_6, BEOL2_6 and BEOL3_6. The naming convention is $\{$*BEOL stack option name*$\}$_$\{$*total number of layers*$\}$. A higher K_{th} value for a particular BEOL stack means that the BEOL stack has a higher routing capacity to sustain more perturbed placements (i.e., placements with more hotspots).

K_{th} versus routing resources. We study the ranking of BEOL stack options with respect to K_{th}. Figure 4.30 shows K_{th} values versus routing resource values, measured using three commercial routers ($R1$, $R2$ and $R3$). The figure shows a positive correlation between K_{th} and routing resource values for all the three routers.[57]

We find that there are several points which show a reversed correlation between K_{th} and routing resources, i.e., a smaller routing resource point shows a higher K_{th}. Also, there are points which have different K_{th} values even though the corresponding routing resource values are the same. This result

[56] In our experiments, K_{th} varies by ≤ 2 across any set of three trials.

[57] To avoid unnecessary flow complications, we report post-route #DRVs out of the routing tool used. We separately verify that routing tools $R1$, $R2$ and $R3$ report the same number of DRVs for a given routed DEF file.

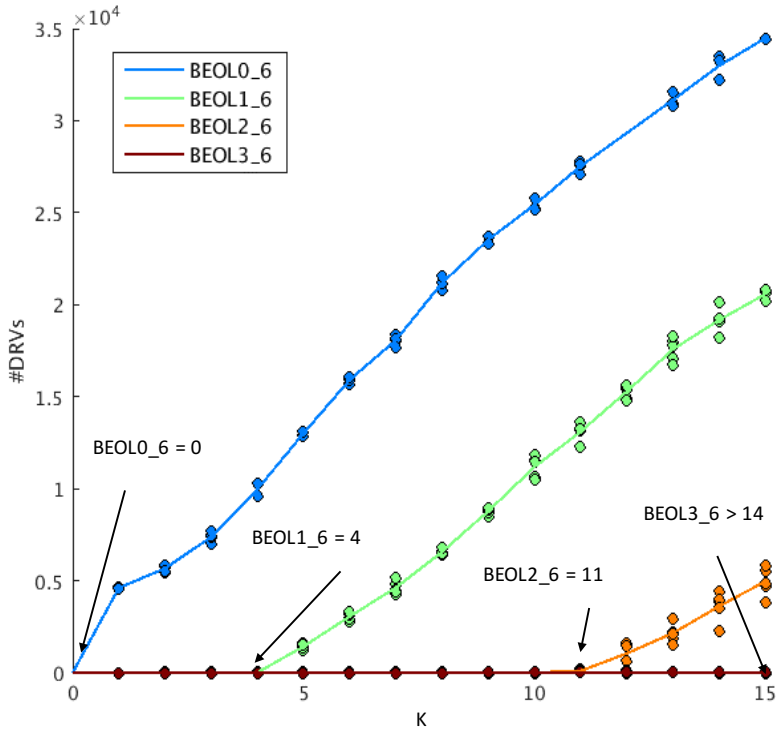


Figure 4.29: The number of DRVs versus K. This result is from a mesh-like placement implemented with 5000 AOI21 cells, and row utilization of 90%.

suggests that "counting routing tracks" may not be an accurate estimation of the routing capacity of a BEOL stack option.[58] Indeed, intuition tells us that measurement of the routing capacity is nontrivial, since gear ratio of metal pitches and via blockages affect routability. Additionally, there could be effects of 'height' of layers – lower metal layers would be more valuable than upper metal layers due to vias. Simply counting routing tracks does not account for such impacts on routability.

BEOL stack options with same routing resource. Since we notice that routing resource may not be a good indicator of routing capacity, we further analyze different BEOL stack options that have the same routing resource value. As $R3$ is an older release of a commercial router, in subsequent experiments below we focus on two routers, i.e., $R1$ and $R2$, which are essentially the latest releases available of two commercial routers. We group BEOL stack options according to routing resources, i.e., all BEOL stack options in each group have the same routing resource value that corresponds to the group. Figure 4.31 and Table 4.9 show K_{th} values for different BEOL stack options with the same routing resource values.

[58]Nevertheless, total track count has been used (and is still used) as a routing capacity metric in industry, and it is an important factor that decides business and technology strategic plans for patterning technology and equipment manufacturing [141].

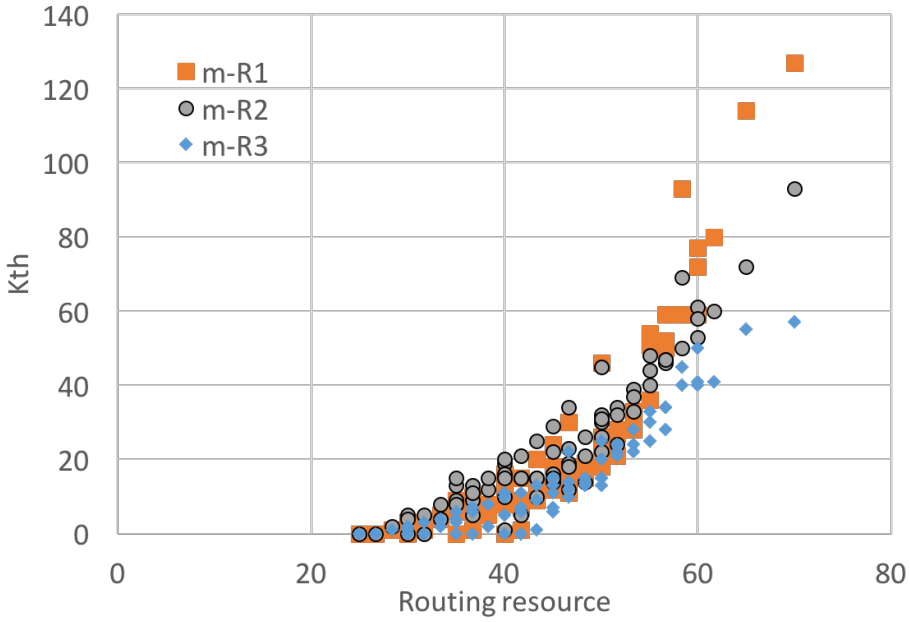


Figure 4.30: K_{th} values versus routing resource with three commercial routers ($R1$, $R2$ and $R3$). The results are extracted using mesh-like placements implemented with 5000 AOI21 cells and 90% row utilization. Each dot corresponds to a BEOL stack option.

The results of six types of implementations, e.g., (i) mesh-like placement with $R1$ (m-$R1$), (ii) mesh-like placement with $R2$ (m-$R2$), (iii) cell width-regularized placement with placer $P1$ and router $R1$ (r-$P1$-$R1$), (iv) placer $P2$ and router $R1$ (r-$P2$-$R1$), (v) placer $P1$ and router $R2$ (r-$P1$-$R2$), and (vi) placer $P2$ and router $R2$ (r-$P2$-$R2$), are reported in the figure. For cell width-regularized placements, placements change depending on BEOL stack options, since BEOL stack options affect placements due to in-built trial routing mechanisms inside the placement tool.

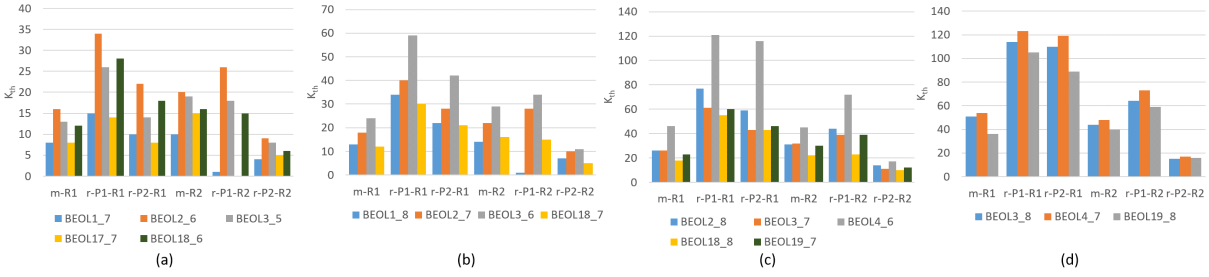


Figure 4.31: K_{th} values for (a) Group 1, (b) Group 2, (c) Group 3 and (d) Group 4 of BEOL stack options in Table 4.9.

Figure 4.32 shows the correlations of the rank-ordering of BEOL stack options (in increasing order

Table 4.9: K_{th} values for groups of BEOL stack options having the same routing resource. The routing resource (T) value for each group is shown in the first column. K_{th} values are measured based on six types of implementations: (i) m-$R1$; (ii) r-$P1$-$R1$; (iii) r-$P2$-$R1$; (iv) m-$R2$; (v) r-$P1$-$R2$; and (vi) r-$P2$-$R2$.

Group (T)	**Name**	**(i)**	**(ii)**	**(iii)**	**(iv)**	**(v)**	**(vi)**
Group 1 (40)	BEOL1_7	8	15	10	10	1	4
	BEOL2_6	16	34	22	20	26	9
	BEOL3_5	13	26	14	19	18	8
	BEOL17_7	8	14	8	15	0	5
	BEOL18_6	12	28	18	16	15	6
Group 2 (45)	BEOL1_8	13	34	22	14	1	7
	BEOL2_7	18	40	28	22	28	10
	BEOL3_6	24	59	42	29	34	11
	BEOL18_7	12	30	21	16	15	5
Group 3 (50)	BEOL2_8	26	77	59	31	44	14
	BEOL3_7	26	61	43	32	39	11
	BEOL4_6	46	121	116	45	72	17
	BEOL18_8	18	55	43	22	23	10
	BEOL19_7	23	60	46	30	39	12
Group 4 (55)	BEOL3_8	51	114	110	44	64	15
	BEOL4_7	54	123	119	48	73	17
	BEOL19_8	36	105	89	40	59	16

of K_{th}) between mesh-like placement results and cell width-regularized placement results. We observe that the rank-ordering based on mesh-like placement and the rank-ordering based on cell width-regularized placements are well-correlated. The correlation coefficients of the rank-orderings of BEOL stacks between the six types of implementations are all larger than 0.9.

We summarize our observations from the results as follows.

- BEOL stack options with the same routing resources show different K_{th} values. This suggests that the routing resource (i.e., sum of routing track counts) alone is not sufficient to quantify the routing capacity of given BEOL stack options.

- The correlation of the rank-orderings of BEOL stack options between cases m-$R1$ and m-$R2$ is high (i.e., correlation coefficient > 0.9). This suggests at least a possibility that the rank-ordering of BEOL stack options is quasi-universal across different routers for the same placements. (From our studies, we conjecture that such quasi-universality holds regardless of the tool that produced the given placement.)

- As shown in Figure 4.32, the correlation of the rank-orderings of BEOL stack options between cases m-$R1$, r-$P1$-$R1$ and r-$P2$-$R1$ is high (i.e., all correlation coefficients > 0.9). Also, the correlation of the rank-orderings of BEOL stack options between cases m-$R2$, r-$P1$-$R2$ and r-$P2$-$R2$ is high (i.e., all correlation coefficients > 0.9). This could indicate that at least in this routability-centric evaluation where other constraints such as timing, power and noise are not considered, the folklore gap between artificial and real benchmarks – in terms of ability to provide insight into CAD heuristic performance – may not be as significant as previously believed.

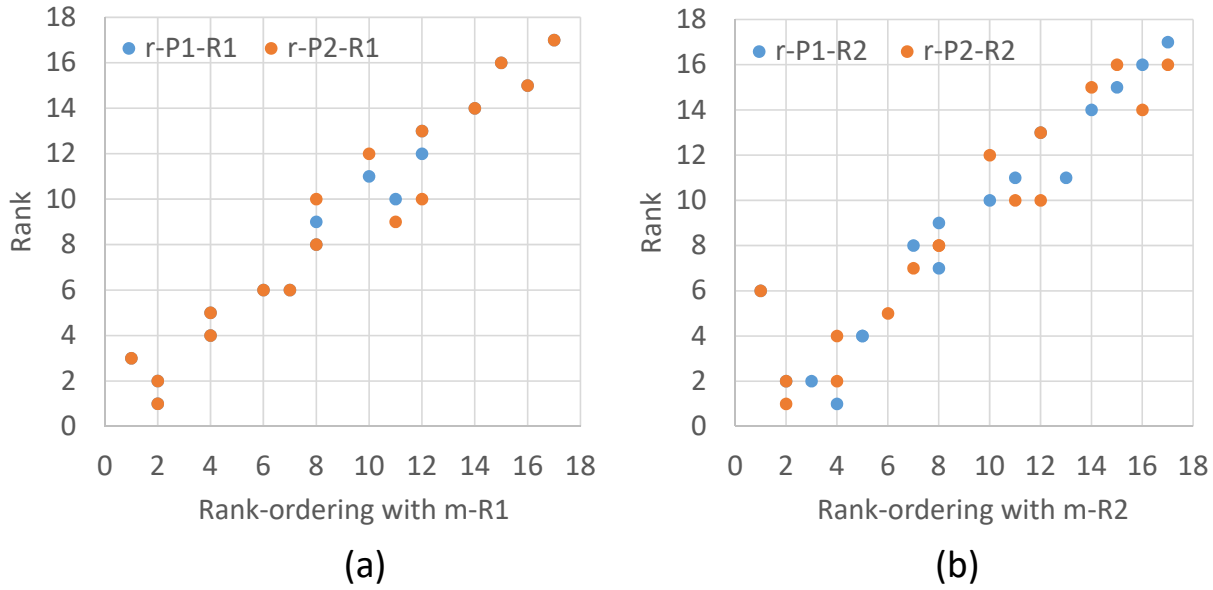


Figure 4.32: Correlations of the rank-ordering of BEOL stack options (in increasing order of K_{th}) between mesh-like placement results and cell width-regularized placement results: (a) $R1$ and (b) $R2$.

Expt2: Comparison of BEOL Stack Options

In this experiment, we seek to understand further the implications of the results from our framework, and which BEOL stack options can better apply a given amount of routing resource. Here, we regard the BEOL stack that offers the larger routing capacity (i.e., having the larger K_{th} in our framework) among all of the candidate BEOL stacks as the better BEOL stack option. First, we study correlation between K_{th} and maximum achievable utilization, in order to understand the impact of BEOL stack options on area. Next, we analyze BEOL stack options with the same K_{th} to derive additional understanding of the

height-dependent routing capacity of metal layers. Finally, we study 1D and 2D routing to answer the question "On which layer(s) is it most valuable to enable bidirectional routing?".

Correlation between K_{th} and maximum achievable utilization. To study the correlation between K_{th} and area, we run P&R with different initial row utilization values and see if there are DRVs after routing. We implement cell width-regularized placements (AES) with bloated cells and sweep the initial row utilization from 45% to 75%.[59] The BEOL stack options in Group 1 are tested with unidirectional routing, placer $P1$ and router $R1$ (r-$P1$-$R1$ in Figure 4.31). We record the maximum achievable utilization values such that #DRVs $<$ 150. The placements are perturbed with K = 20 to make routing harder. Figure 4.33 shows the K_{th} values obtained from cell width-regularized placements (r-$P1$-$R1$) in blue bars (left y-axis) and the corresponding maximum achievable (initial) row utilization values in orange trace (right y-axis). We observe that a BEOL stack option with a higher K_{th} achieves a higher maximum achievable utilization.

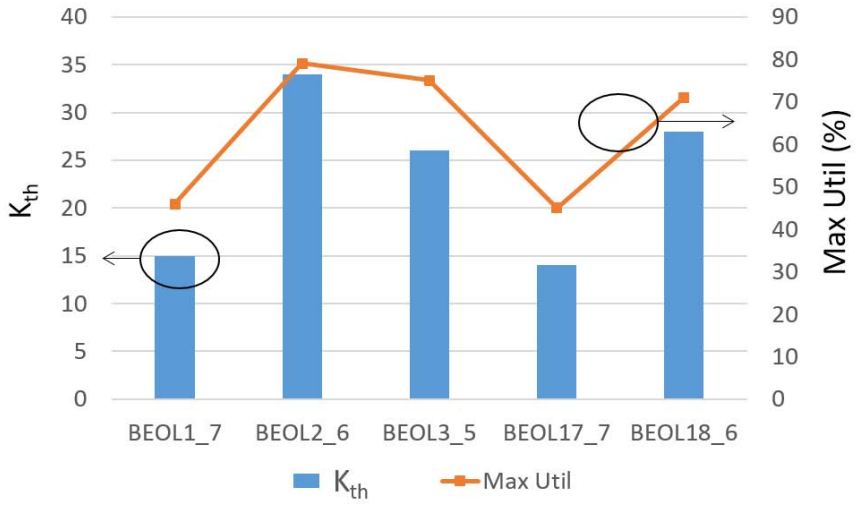


Figure 4.33: The K_{th} values obtained from cell width-regularized placements versus maximum achievable (initial) row utilization values. The BEOL stack options in Group 1 are tested using the AES design and $R1$. We record the maximum achievable utilization values such that #DRVs $<$ 150 (shown in the orange trace).

Analyses of BEOL stack options with same K_{th}. Figure 4.34 shows the results of m-$R1$ mesh-like placements for subsets of BEOL stack options with same K_{th} values. Figures 4.34(a), (b), (c) and

[59] For row utilizations $>$ 75%, the results do not change since bloated standard cells introduce porosities in placements that we cannot control using initial row utilizations. For example, for 8T cell masters, the average width increase is 2.22 placement sites, which means that every cell effectively has $\sim$2 placement sites of embedded, i.e., internal, whitespace.

(d) show the results of BEOL stack options with K_{th} = 4–6, K_{th} = 9–11, K_{th} = 14–16 and K_{th} = 19–21, respectively. Since K_{th} indicates routing capacity, BEOL stack options with the same K_{th} values are regarded as equivalent with respect to routing capacity. Example findings from m-$R1$ results are as follows.

- The added or incremental routing capacity of a layer depends on the height of the layer according to the m-$R1$ results, as one would expect. For example, in Figure 4.34(a), one 1× layer on M3 is equivalent to two 2× layer on M5 and M6 (BEOL12_6 and BEOL13_5); or, one 1× layer on M3 and one 2× layer on M5 are equivalent to two 1.5× layers on M3 and M5 (BEOL8_6 and BEOL17_6). In Figure 4.34(b), two 1× layers on M3 and M4 are equivalent to four 1.5× layers on M3, M4, M5 and M6 (BEOL2_5 and BEOL21_7).

- We also observe that if the numbers of 1× and 1.5× layers are sufficient, higher metal layers do not significantly affect routing capacity. In Figure 4.34(b), if the number of 1× and 1.5× layers ≥ 6, additional layers do not help.

- Last, the data indicate that using more 1× layers reduces the required total number of layers for a given routing capacity. Obviously, added dimensions to our study, such as signal integrity performance and/or power delivery, are directions for follow-on research.

1D versus 2D routing. We study the question "On which layer(s) is it most valuable to enable bidirectional routing?". Here, we use *1D routing* to indicate unidirectional routing where routing in the non-preferred direction is not allowed. We use *2D routing* to indicate bidirectional routing where routing segments in both horizontal and vertical directions can exist on the given layer. We use the same metal pitch for both directions for 2D routing. We compare K_{th} of various BEOL options with six 1× 1D layers (e.g., BEOL4_6) and one 2D routing layer enabled using the m-$R1$ implementation. Table 4.10 shows the layer configuration of each BEOL option and the corresponding K_{th} value. We note that the routing resource of each option in this experiment is the same since all metal layers are 1× layers. However, each option has different horizontal and vertical resources. 1D, 2D-B and 2D-D have three horizontal routing layers and two vertical routing layers. And, 2D-A and 2D-C have two horizontal routing layers and three vertical routing layers. All BEOL options with one 2D routing layer show smaller K_{th} compared to the

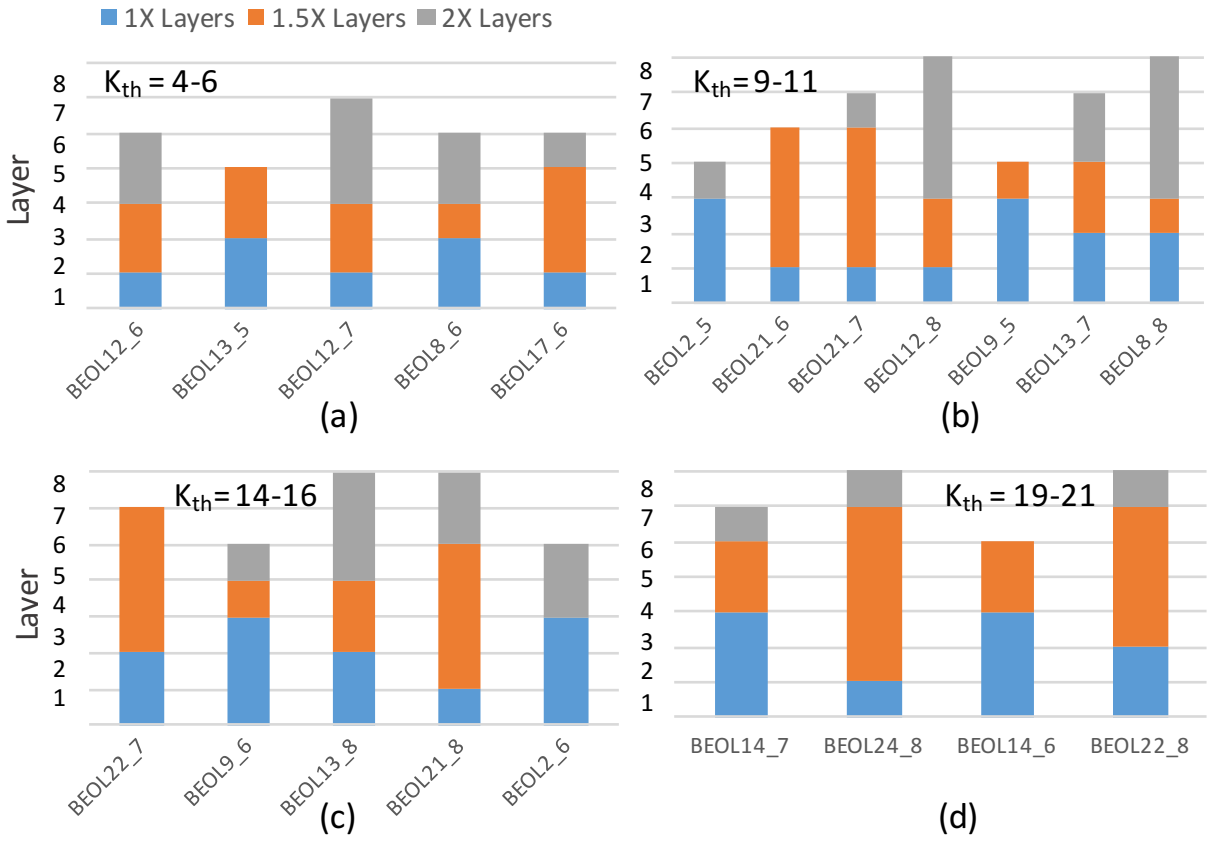


Figure 4.34: Results of m-$R1$, shown for subsets of BEOL stack options that have the same K_{th} values. (a) BEOL stack options with K_{th} = 4–6; (b) BEOL stack options with K_{th} = 9–11; (c) BEOL stack options with K_{th} = 14–16; and (d) BEOL stack options with K_{th} = 19–21.

1D BEOL option. This may indicate that having two 1D layers is always better than having one 2D layer. Even though the routing resource (measured by pure track count) is the same in both cases, routing rules can limit the utilization of the given routing resource in a 2D routing layer.

By comparing between 2D options, we observe that it may be more valuable to enable 2D routing on lower metal layers; this can be seen by comparing 2D-B versus 2D-D, and 2D-A versus 2D-C. We may also infer that horizontal routing resources are more valuable than vertical resources, for our nearly-square blocks comparing 2D-A versus 2D-B, or 2D-C versus 2D-D. That is, BEOL options with more horizontal routing resources show higher K_{th} values (i.e., better routing capacity). We do not claim to be the first to observe such correlations, and our results are each specific to a given enablement. This being said, ours is the first *framework* for quantifying such assessments in a general, design-agnostic manner.

Table 4.10: K_{th} results of various 2D options.

Option	M2	M3	M4	M5	M6	K_{th}
1D (BEOL4_6)	1D	1D	1D	1D	1D	46
2D-A	2D	1D	1D	1D	NA	13
2D-B	1D	2D	1D	1D	NA	14
2D-C	1D	1D	2D	1D	NA	12
2D-D	1D	1D	1D	2D	NA	13

Expt3: Robustness of the Rank-Ordering of BEOL Stack Options

We now show further experimental results to support the robustness of our observed quasi-universal rank-ordering of BEOL stack options across mesh-like placements and cell width-regularized placements, and two routers.

Various mesh-like placement configurations. We perform experiments for various mesh-like placements implemented with different configurations. Note that the baseline is the AOI-mesh in Table 4.7. The different configurations for mesh-like placements are summarized as follows.

- Different total numbers of instances (#Instances = 5000, 15000 and 20000)
- Different standard cell types (NAND2 and AOI21)
- Different row utilizations (70%, 80% and 90%)
- Different pin alignment
- Different routing direction (1D versus 2D)
- Different standard cells (8T and 12T)

Figure 4.35 shows K_{th} values for five BEOL stack options, measured with various mesh-like placements with different configurations. The five BEOL stack options have the same routing resources, $T = 40$ (Group 1 in Table 4.9). Our findings from the results are as follows.

- Figure 4.35(a) shows the results of two mesh-like placements implemented with AOI21 (three-input cell) and NAND2 (two-input cell). The width values of AOI21 and NAND2 are 1.088μm and 0.952μm, respectively. The NAND2 case has higher K_{th} values. This is because the NAND2-based

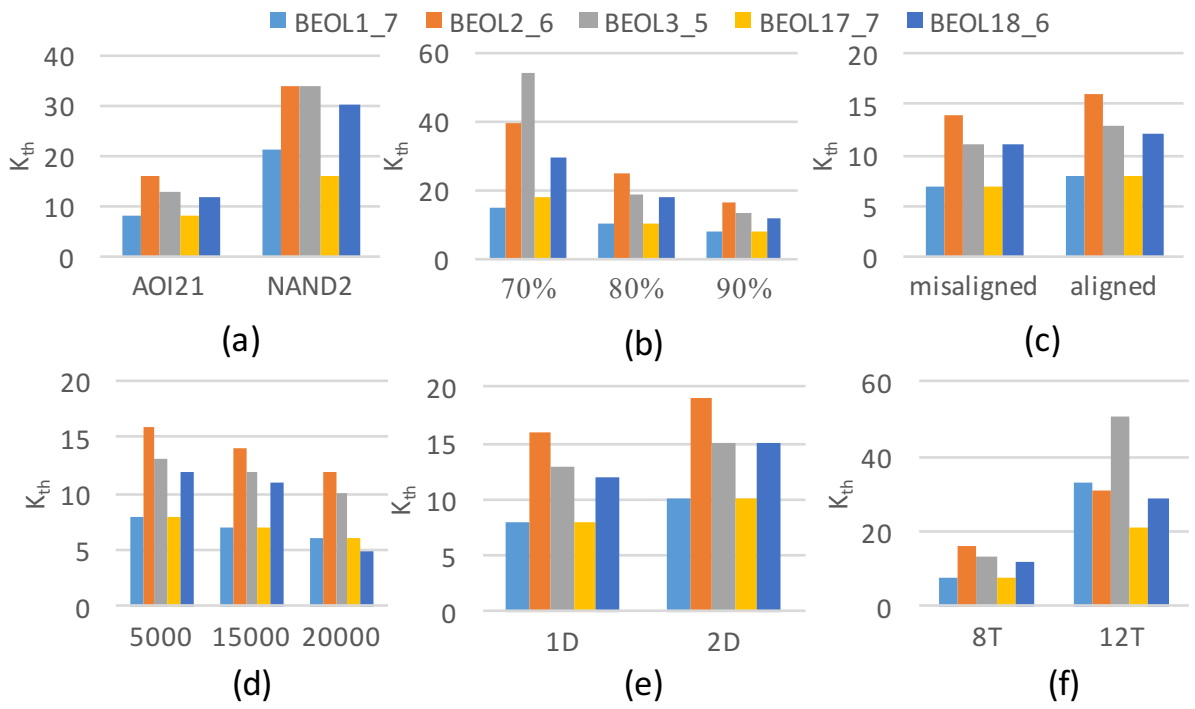


Figure 4.35: K_{th} of mesh-like placements with various configurations. (a) Different cell types: AOI21 (three-input cell) and NAND2 (two-input cell); (b) different row utilizations: 70%, 80% and 90%; (c) different pin alignments; (d) different total number of instances: 5K, 15K and 20K; (e) 1D and 2D routing; and (f) 8T and 12T cells.

implementation has a lower net degree. However, the rank-ordering of BEOL stack options with respect to K_{th} values is the same as that of the AOI21 case.

- Figure 4.35(b) shows the results of m-$R1$ with different row utilizations, 70%, 80% and 90%.[60] As expected, placements with lower utilizations achieve higher K_{th}. The rank-ordering of BEOL stack options with respect to K_{th} is the same for 80% and 90% cases, but there is a deviation for 70% case. (i.e., reversed ordering of BEOL3_5 and BEOL2_6).

- Figure 4.35(c) shows the impact of different pin alignments. In a mesh-like placement, pins

[60] We observe that in the implementation with 70% utilization, due to rounding effects the space between two horizontally adjacent cells can vary (three or four placement sites). Such cases, where cells are not distributed uniformly, lead to more DRVs in the router outcome. To make a mesh-like placement with uniform distribution, the row utilizations we can implement are limited to $w/(w+s)$, where w and s are the cell width and the spacing between cells, respectively. Since w, s are integer (in placement sites) and since $w = 8$ in the experiment, we can implement row utilizations 72.7%, 80.0% and 88.9% with $s = 3, 2, 1$, respectively. To implement an exact 70% utilization, we would need to use multiple values of s, which would result in non-uniform mesh-like placements. Therefore, we use 72.7%, 80.0% and 88.9% utilizations for 70%, 80% and 90%, respectively, to make a fairer comparison among the three cases.

of standard cells are aligned unless we apply different offsets for each placement row. For the "misaligned" case, we add a different offset for each placement row to create vertical misalignment between cells in adjacent placement rows. The result suggests that the impact of pin alignment is negligible.

- Figure 4.35(d) shows the impact of different numbers of instances, i.e., 5000, 15000 and 20000. As the number of instances increases, K_{th} slightly decreases. This result suggests that design size is related to K_{th} (See Section 4.3.5, below). K_{th} of BEOL18_6 is relatively lower in the 20000 case, and this results in a different rank-ordering of BEOL stack options. However, except for BEOL18_6, the rank-ordering of BEOL stack options remains consistent.

- Figure 4.35(e) shows results of 1D and 2D routing, where bidirectional routing is enabled for all metal layers for the 2D routing. (Note that this is a different experiment from **1D versus 2D routing** above, where bidirectional routing is enabled for only one layer.) K_{th} is higher with 2D routing, which suggests routing capacity of 2D routing is larger. The rank-ordering of BEOL stack options remains the same.

- Figure 4.35(f) shows results of 8T and 12T cells. In this experiment, we use 8T and 12T cells with the same width. K_{th} is higher with 12T cell, as one would expect. The rank-ordering of BEOL2_6 and BEOL3_5 is reversed with 12T cell, as compared to other configurations.[61]

In summary, the rank-ordering of the BEOL stack options with respect to K_{th} does not change significantly across different mesh-like placements with a wide range of configurations. However, there do exist deviations across configurations (track height, utilization, routing directionality), indicating that no one configuration perfectly represents all other possible configurations. Thus, it would be important for designers to select a proper set of configurations to reflect the properties of target designs.

Cell width-regularized placement with standard cell variants. We perform similar experiments with cell width-regularized placements using more standard cell variants[62], for five BEOL stack options

[61] We have applied multiple runs (> 10) to further remove noise due to randomness of the sequence of neighbor-swaps, but have found results to be consistent (i.e., stable).

[62] We use 8T and 12T cells bloated to have different widths (i.e., eight and 10 placement sites). The width of a flip-flop is 23 placement sites.

in Group 1. Figure 4.36 shows K_{th} values of the BEOL stack options in Group 1 for four cases among the combinations of two heights (8T and 12T), two widths (eight and 10 placement sites) and two routing directions (1D and 2D). I.e., (i) 8T bloated cells (width=8) and unidirectional routing (*R-8T-B1-1D*), (ii) 8T bloated cells of a wider width (width=10), and unidirectional routing (*R-8T-B2-1D*), (iii) 8T bloated cells (width=8) and bidirectional routing (*R-8T-B1-2D*) and (iv) 12T bloated 12T cells (width=8) and unidirectional routing (*R-12T-B1-1D*). The results are obtained using $P1$ and $R1$ (Figure 4.36(a)), and $P2$ and $R2$ (Figure 4.36(b)).[63] We observe that R-8T-B2-1D shows higher K_{th} values than R-8T-B1-1D. This may be due to larger spacings between pins in the R-8T-B2-1D case. We also see that R-8T-B1-2D shows slightly higher K_{th} values when compared to R-8T-B1-1D. We observe that using 12T cells increases routing capacities dramatically (R-12T-B1-1D).

Placements generated by a standard SP&R flow. We support our observed routing capacity-based ordering of BEOL stacks using placements that are implemented by a standard SP&R flow. Specifically, we implement placements for the AES and VGA testcases using a full set of library cells without any modification, and apply our neighbor-swap-based approach. As noted in Footnote 50, to maintain placement legality we check whitespace after every neighbor-swap operation between placement rows. If the row utilization with the updated whitespace exceeds a pre-defined sum of initial row utilization + margin, we revert the neighbor-swap operation. For AES and VGA, we use 50% and 45% for initial row utilization, respectively, and we use 1% margin for both designs.

Algorithm 13 gives details of the procedure for perturbing real-cell-based real placements. We initialize the number of neighbor-swaps performed (num_swap) in Line 1. We iteratively perform neighbor-swaps until the total number of neighbor-swaps reaches the target (calculated from the given N^2 and K). For a neighbor-swap, we randomly select target cell ($cell$) and direction (dir) (north, south, east and west) in Lines 3 and 4. This determines the neighbor cells of $cell$ (Line 5). In Line 6, the target cell and its neighbor cell are swapped, and row utilizations are updated (Line 7). We increment num_swap in Line 8. If the neighbor-swap is performed between rows and the widths of the target cell and its neighbor cell are different (Line 9), we check that updated row utilizations do not exceed a pre-defined limit (Line

[63] We could not obtain results for case (iv) with $P2$ and $R2$, since $R2$ could not produce DRV-clean results for $K = 0$ with 12T cells.

10). If this check fails, we revert the neighbor-swap operation (Lines 11 and 12).

The use of real cell widths leads to less-gradual perturbation due to the effect of placement legalization before routing. The results of the non-bloated-cell-based implementation are shown in Figure 4.37 (*R-8T-NB-1D-AES* and *R-8T-NB-1D-VGA*). *R-8T-B1-1D-AES* is given as a reference (*R-8T-B1-1D* in Figure 4.36). We see that indeed, K_{th} values of *R-8T-NB-1D-AES* and *R-8T-NB-1D-VGA* are dramatically smaller than those of *R-8T-B1-1D-AES*. Although our focus is on the rank-orderings of BEOL stack routing capacities based on K_{th}, as opposed to the magnitudes of K_{th} values, this experiment clearly shows a gap between cell width-regularized placements and placements generated within production SP&R flows. We emphasize that the AES-derived, width-regularized testcase is a compromise between real and synthetic, due to the cell bloating. And, further understanding of the significance of the cell bloating used in our studies of the PROBE methodology remains an important direction for future research. In terms of the rank-ordering, the result of *R-8T-NB-1D-AES* remains the same as the reference, while there is a deviation (BEOL2_6) in the *R-8T-NB-1D-VGA* case. This data point may indicate that the rank-ordering has additional dependencies on netlist structure and size.

Algorithm 13 Placement perturbation

Input: an input placement, total number of instances N^2, target K, target utilization U, utilization margin M
Output: a perturbed placement

1: $num\_swap \leftarrow 0$
2: **while** $num\_swap < N^2 \cdot K$ **do**
3: $cell \leftarrow RandomlySelectCell$
4: $dir \leftarrow RandomlySelectDirection$
5: $neighbor\_cell \leftarrow GetNeighborCell(cell, dir)$
6: Swap cell, neighbor_cell
7: Update row utilization
8: $num\_swap \leftarrow num\_swap + 1$
9: **if** $((dir == north)\ ||\ (dir == south))$ && $w_{cell} \neq w_{neighbor\_cell}$ **then**
10: **if** row utilization $> U + M$ for any row **then**
11: Swap cell, neighbor_cell (revert)
12: $num\_swap \leftarrow num\_swap - 1$
13: **end if**
14: **end if**
15: **end while**

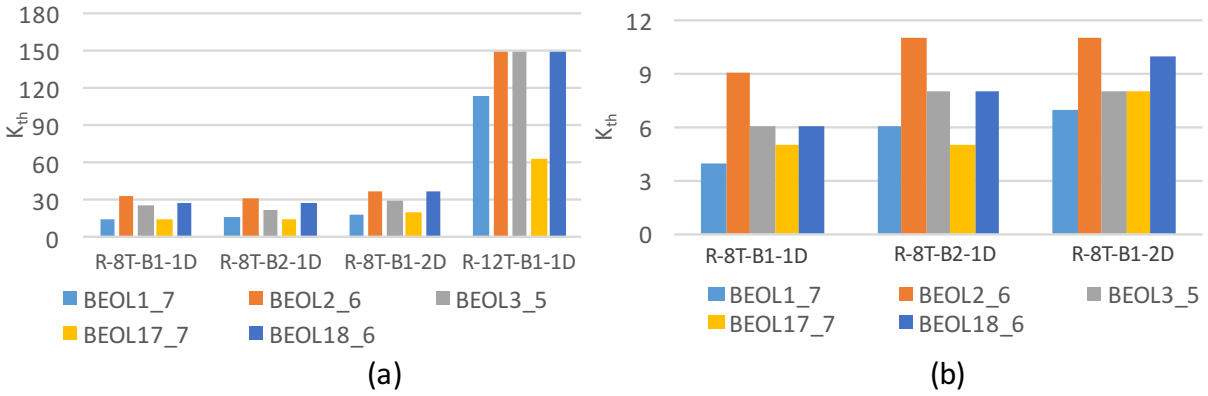


Figure 4.36: K_{th} values of the BEOL stack options in Group 1 for various cases: (i) R-8T-B1-1D, (ii) R-8T-B2-1D, (iii) R-8T-B1-2D and (iv) R-12T-B1-1D. The results are obtained (a) using ($P1$, $R1$) and (b) using ($P2$, $R2$).

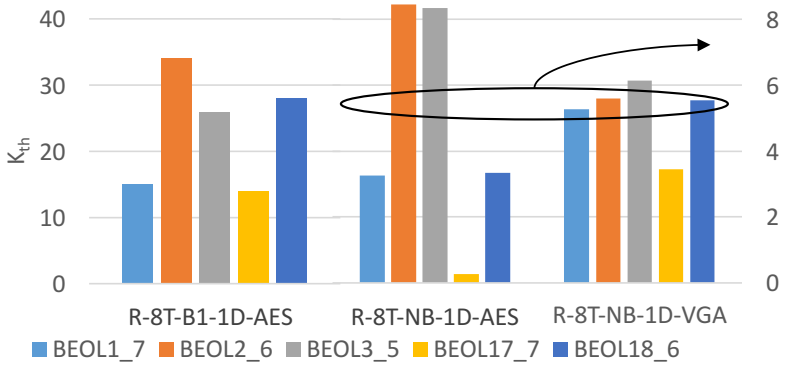


Figure 4.37: K_{th} values of the BEOL stack options in Group 1 for AES and VGA placements with non-bloated 8T cells (*R-8T-NB-1D-AES* and *R-8T-NB-1D-VGA*). *R-8T-B1-1D-AES* is shown for a reference. The results are obtained using ($P1$, $R1$).

4.3.4 Additional Study of Routing Hotspot and Routing Failure

The studies above focus on ranking of BEOL stack options based on fixed-size designs and use of the K_{th} criterion as an indicator of routing capacity. The K_{th}-based BEOL stack ordering is shown empirically to be stable across a number of factors - routing tool, netlist topology, utilization, porosity, layer directionality, etc. Notably, Figure 4.35(d) suggests that the K_{th}-based stack ordering is independent from design size, even as the K_{th} values themselves change with design size.

By necessity, our studies involve small netlists, which raises the important question of how to extend insights to when designs are large. In this section, we provide additional studies of (i) the size

and placement quality of a routing hotspot, and (ii) the impact of design size on routing failure. These provide context for the preceding experimental results: (i) routing failure (in hotspots) is a function of both hotspot size and placement suboptimality (K_{th}), and (ii) the observed change of K_{th} with design size (Figure 4.35(d)) matches the outcomes of a more quantitative analysis.

Routing Hotspots

Routing failure is caused by local routing hotspots in many cases. However, not every routing hotspot contributes equally to routing failure. Figure 4.38 shows hotspots of two different sizes, along with post-route DRVs (white crosses). Both hotspots are generated with the same number $K \times S^2$ of neighbor-swaps (where S is the dimension of the $S \times S$ hotspot, and defines the size of the hotspot). We observe that while the two hotspots have the same K, the numbers of DRVs are different, and hotspot1 does not contribute much to DRVs. This example suggests that the size of routing hotspots is another key factor - in addition to K - that determines routing failure. Thus, in this section, we further empirically study various sizes of routing hotspots.

In mesh-like placements, we generate routing hotspots with various locations and sizes. Specifically, we vary the size of a hotspot (i.e., by performing $K \times S^2$ random neighbor-swaps within a specific subregion). An $S \times S$ hotspot is a subregion with S columns and S rows. We study various (S, K) pairs and record #DRVs. For each (S, K) pair, we generate 10 random data points. Figure 4.39 shows a contour map that indicates routability for various (S, K) pairs. The x-axis shows K normalized by S^2 because there are $O(S^2)$ total edges, and the y-axis shows S. The solid (resp. dotted) line is the contour based on average (resp. maximum) numbers of DRVs (of 10 trials for the corresponding (S, K) pair), where the upper shaded regions correspond to routing failures. The figure shows that (i) if the size of hotspots (S) is smaller than a certain value, then $K_{th} = \infty$; and (ii) the K_{th} values increase as the size of hotspots decreases.

Impact of Design Size on Routing Failure

Design size is an important factor (along with routing algorithms, BEOL stack options, netlist topology, placement utilization, etc.) that affects routing difficulty. To better understand the connection

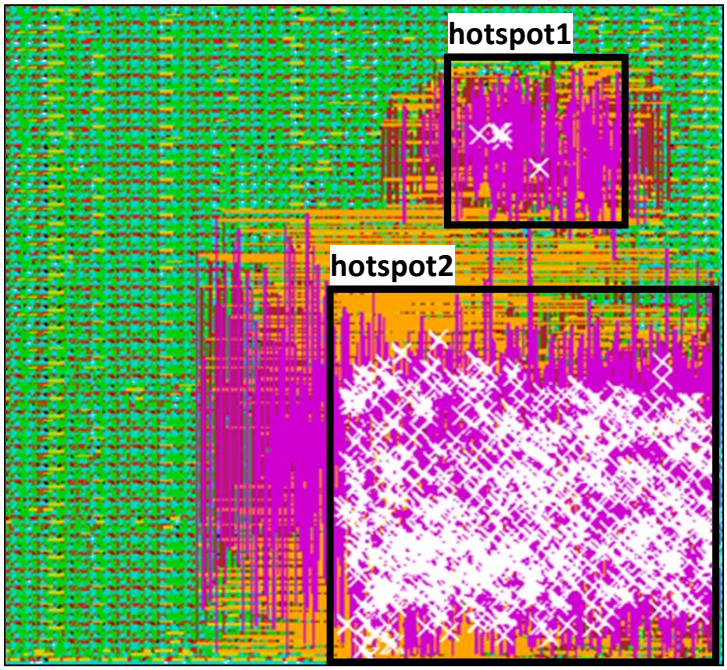


Figure 4.38: A routed layout with two hotspots. DRVs are indicated by white crosses. Other colors (green, pink, orange) indicate metal layers. Although each hotspot is perturbed by the same K ($\times S^2$, where S is the dimension of the $S \times S$ hotspot), hotspot1 ($S = 16$) produces noticeably fewer DRVs than hotspot2 ($S = 33$).

between design size and routing difficulty that is seen in Figure 4.35(d), we now provide a more quantitative study of the impact of design size on the probability of routing failure. That is, given similar placement quality (measured by the size-normalized number of neighbor-swaps from a mesh-like placement), we examine the relation between design size and the probability of routing failure.

As reviewed in Section 4.3.1, many metrics have been proposed to estimate routability, such as pin density [15], pin shape [148], wire density [142], Rent exponent [163], net range [108] and the number of incoming/outgoing edges [24]. We study two metrics that have been closely related to routing failures in the recent work of [24]. The first metric is the *sum of edge distances* (ED), i.e., the sum of half-perimeter wirelengths of nets corresponding to lattice edges initially in a given window. (An $S \times S$ window of the initial square mesh contains $2 \cdot S \cdot (S - 1)$ lattice edges.) The second metric is *crossing count* (CC), which is the total number of nets that *cross* a given local window. In other words, nets having positive-area overlap with a given window are counted in CC (overlap only along the window boundary is not counted). Figure 4.40 shows examples of edge mapping, ED and CC. In Figure 4.40(a), initial lattice edge ((p_1, q_1),

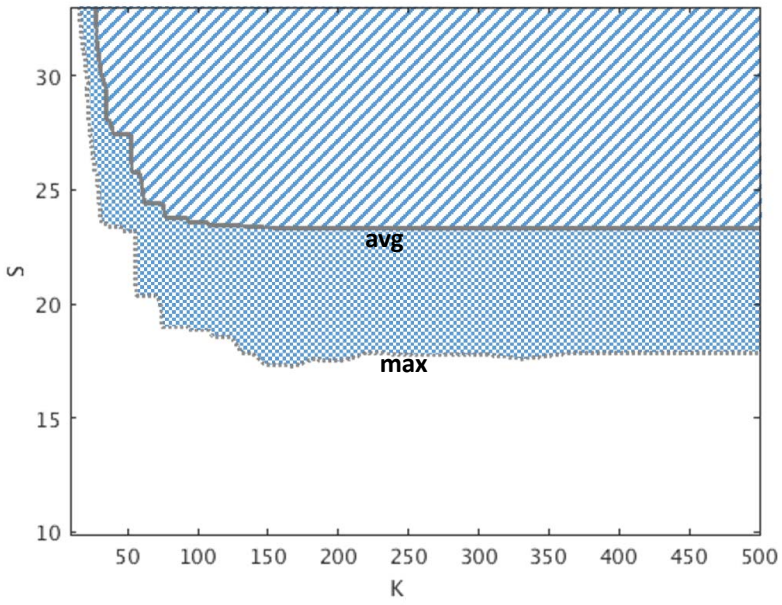


Figure 4.39: A contour map that indicates routability for various (S, K) pairs. The solid line is the contour based on the average number of DRVs, and the dotted line is the contour based on the maximum number of DRVs in 10 trials per each (S, K) pair. Based on the contour lines, the upper shaded regions show where routing fails.

(p_2, q_2)) (vertices (p_1, q_1) and (p_2, q_2) are adjacent in the initial mesh) is *mapped to* ((p_3, q_3), (p_4, q_4)) after neighbor-swaps. For this edge, the $ED = |p_3 - p_4| + |q_3 - q_4|$. In Figure 4.40(b), in red are net bounding boxes, and in blue is the window, where Nets B and C contribute to CC, but Nets A and D do not.

We perform exponentiation on transition matrices to estimate the ED and CC changes after neighbor-swaps, which further provide an estimation for the probability of routing failures (i.e., ED (or CC) is greater than a threshold ED_{th} (or CC_{th})).

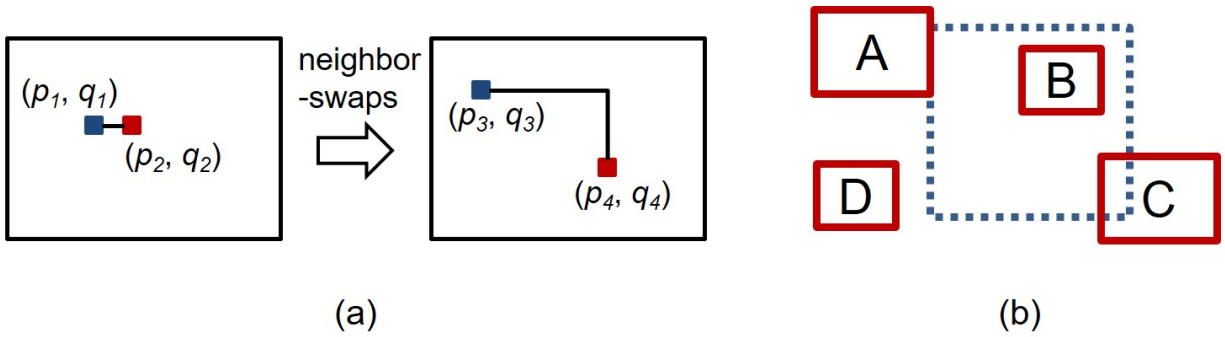


Figure 4.40: Illustrations of edge mapping, ED and CC.

We describe our transition matrix-based estimation as follows. Given a mesh-like placement (where an edge exists between each pair of neighboring instances) with size ($N \times N$), and number of neighbor-swaps ($K \times N^2$), we estimate the expected ED within the hotspot based on a transition matrix (i.e., a matrix used to describe the transition of iterative neighbor-swaps, where $M_T[i][j]$ is the probability that a vertex moves from location i to location j in a single neighbor-swap [10]). Specifically, from a given matrix that is the exponentiation (to the K^{th} power) of a given transition matrix, we compute the ED and CC over a given hotspot region.

Algorithm 14 describes the transition matrix construction. Based on the transition matrix, we are able to estimate the probability of *mapping* an edge ((p_1, q_1), (p_2, q_2)) to another edge ((p_3, q_3), (p_4, q_4)) (i.e., embedded in the matrix as the entry in the $(N^3 \cdot p_1 + N^2 \cdot q_1 + N \cdot p_2 + q_2)^{th}$ row and the $(N^3 \cdot p_3 + N^2 \cdot q_3 + N \cdot p_4 + q_4)^{th}$ column). We then multiply such probabilities by the Manhattan length of the (post-neighbor-swapping) mapped edge to achieve an estimation of the expected edge distance. Lastly, we take the sum over expected edge distances of the lattice edges (i.e., edges between adjacently-placed cells in the initial mesh). In Line 9, we set $M_T[i][i]$ to the difference between $2 \cdot N \cdot (N-1)$ and $rowSum$ so that when we normalize the rows by $2 \cdot N \cdot (N-1)$ (Line 11), every entry will represent a marginal probability of transitioning to that state.[64] However, due to the large size of the transition matrix $O(N^4)$, the runtime complexity of the dense matrix exponentiation is $O(N^{4k})$, where $O(N^k)$ = matrix multiplication complexity. Thus, the complexity of the sparse matrix exponentiation is $O(N^8)$.

To reduce the runtime complexity, we propose an *an approximate calculation approach*, shown in Algorithm 15. The transition matrix in Algorithm 15 is of size $N^2 \times N^2$ as opposed to $N^4 \times N^4$ in Algorithm 14, because it keeps track of the mapped vertex locations as opposed to the mapped edges. Thus, Algorithm 15 assumes that the mapped vertex locations are independent, which is intuitively reasonable given that the expected number of neighbor-swaps for each vertex is still the same. In Algorithm 15, we first create a distance matrix M_D of size $N^2 \times N^2$, where $M_D[N \cdot p_1 + q_1][N \cdot p_2 + q_2]$ is the Manhattan distance between (p_1, q_1) and (p_2, q_2) in the mesh-like placement (Line 1). We then create a transition matrix M_T (Line 2), as presented in Algorithm 16. We exponentiate the transition matrix to the K^{th} power,

[64]The normalization factor is $2N(N-1)$ because this is the total number of events or possible neighbor-swaps that can occur. Thus, in Line 11, we divide the matrix through by $2N(N-1)$ to obtain a transition matrix in which each row sums to 1.

to approximate the resultant placement after K moves (Line 3). Lastly, according to M_E, we calculate the expected edge distance, as summarized in Algorithm 17. The runtime complexities for construction of the distance matrix (Line 1), construction of the transition matrix (Line 2), exponentiation of the transition matrix (Line 3) and calculation of the expected edge distances (Line 4) are respectively $O(N^4)$, $O(N^4)$, $O(N^6 \cdot log(K))$ and $O(N^6)$.[65] Overall, the runtime complexity of our procedure is $O(N^6 \cdot log(K))$.

Algorithm 14 Create transition matrix (exact), M_T

1: $M_T \leftarrow N^4 \times N^4$ zero matrix
2: **for** p_1 := 1 to N, q_1 := 1 to N, p_2 := 1 to N, q_2 := 1 to N, p_3 := 1 to N, q_3 := 1 to N, p_4 := 1 to N, q_4 := 1 to N **do**
3: **if** ((((p_1, q_1) == (p_3, q_3) && isNeighbor((p_2, q_2), (p_4, q_4))) || ((p_2, q_2) == (p_4, q_4) && isNeighbor((p_1, q_1), (p_3, q_3))) || ((p_1, q_1) == (p_4, q_4) && (p_2, q_2) == (p_3, q_3) && isNeighbor((p_1, q_1), (p_2, q_2)) && ((p_1, q_1) $\neq$ (p_2, q_2) || (p_3, q_3) $\neq$ (p_4, q_4)))) **then**
4: $M_T[N^3 \cdot p_1 + N^2 \cdot q_1 + N \cdot p_2 + q_2][N^3 \cdot p_3 + N^2 \cdot q_3 + N \cdot p_4 + q_4] \leftarrow 1$
5: **end if**
6: **end for**
7: **for** i := 1 to N^4 **do**
8: $rowSum \leftarrow \sum_j M_T[i][j]$
9: $M_T[i][i] \leftarrow 2 \cdot N \cdot (N-1) - rowSum$
10: **end for**
11: $M_T \leftarrow M_T/(2 \cdot N \cdot (N-1))$

Algorithm 15 Expected edge distance (approximation)

1: $M_D \leftarrow$ create distance matrix
2: $M_T \leftarrow$ create transition matrix
3: $M_E \leftarrow$ exponentiate transition matrix M_T to the K^{th} power
4: $dist \leftarrow$ calculate expected edge distance
5: **return** $dist$

Similarly, Algorithm 18 describes our approximate calculation of crossing count. Algorithm 16 and Algorithm 19 respectively describe the transition matrix construction and expected crossing count calculation. To calculate CC, in Algorithm 19 Line 4, the variable $count$ contains the expected number of edges, where we add in the probability that ((p_1, q_1), (p_2, q_2)) is mapped to ((p_3, q_3), (p_4, q_4)) by assuming independence and multiplying the probability that (p_1, q_1) is mapped to (p_3, q_3) by the probability that (p_2, q_2) is mapped to (p_4, q_4). Furthermore, for every pair of undirected edges, we add the probability of that

[65] We use Python np.linalg library to perform matrix exponentiation.

Algorithm 16 Create transition matrix (approximation)

1: $M_T \leftarrow N^2 \times N^2$ zero matrix
2: **for** p_1 := 1 to N, q_1 := 1 to N, p_2 := 1 to N, q_2 := 1 to N **do**
3: **if** isNeighbor((p_1, q_1), (p_2, q_2)) **then**
4: $M_T[N \cdot p_1 + q_1][N \cdot p_2 + q_2] \leftarrow 1$
5: **end if**
6: **end for**
7: **for** i := 1 to N^2 **do**
8: $rowSum \leftarrow \sum_j M_T[i][j]$
9: $M_T[i][i] \leftarrow 2 \cdot N \cdot (N-1) - rowSum$
10: **end for**
11: $M_T \leftarrow M_T/(2 \cdot N \cdot (N-1))$

Algorithm 17 Calculate expected edge distance

1: $dist \leftarrow 0$
2: **for all** edges ((p_1, q_1), (p_2, q_2)) **do**
3: **for** i := 1 to N^2, j := 1 to N^2 **do**
4: $dist$ += $M_E[N \cdot p_1 + q_1][i] \cdot M_E[N \cdot p_2 + q_2][j] \cdot M_D[i][j]$
5: **end for**
6: **end for**
7: **return** $dist$

mapping four times.[66] Specifically, we count the edge mapping of (v_1, v_2) to (v_3, v_4) once for each of the two edge permutations of (v_1, v_2) and once for each of the two mapped edge permutations of (v_3, v_4). So, we count this probability $2 \cdot 2 = 4$ times; we then divide by 4 to eliminate double counting (Line 7). We note that for a large K value, the approximated edge distance and crossing count should be off by a factor of $(N^2)/(N^2-1)$ with respect to the exact values. This is because the approximation algorithms count the set of degenerate edges.[67]

Algorithm 18 Expected crossing count approximation

1: $M_T \leftarrow$ create transition matrix
2: $M_E \leftarrow$ exponentiate transition matrix M_T to the K^{th} power
3: $count \leftarrow$ calculate expected crossing count
4: **return** $count$

Based on the transition matrix exponentiation, we estimate the probabilities of $ED \geq ED_{th}$ and

[66] Here, an undirected edge is an unordered pair of vertices (e.g., edge ((p_1, q_1), (p_2, q_2)) in Figure 4.40(a)). In our constructed mesh placement, there can be at most one net between two instances, and we therefore treat each net as an undirected edge in our analyses.

[67] There are $(N^2)(N^2-1)$ total edges, and there are $(N^2)(N^2)$ ordered pairs of vertices, which the approximation algorithm counts. Therefore, for large K, we expect to be off by around a factor of $(N^2)/(N^2-1)$ with respect to the exact values.

Algorithm 19 Calculate crossing count

1: $count \leftarrow 0$
2: **for all** edge $((p_1, q_1), (p_2, q_2))$ **do**
3: **for all** mapped edge $((p_3, q_3), (p_4, q_4))$ that overlap with the given $S \times S$ hotspot **do**
4: $count$ += $M_E[N \cdot p_1 + q_1][N \cdot p_3 + q_3] \cdot M_E[N \cdot p_2 + q_2][N \cdot p_4 + q_4]$
5: **end for**
6: **end for**
7: $count \leftarrow count/4$
8: **return** $count$

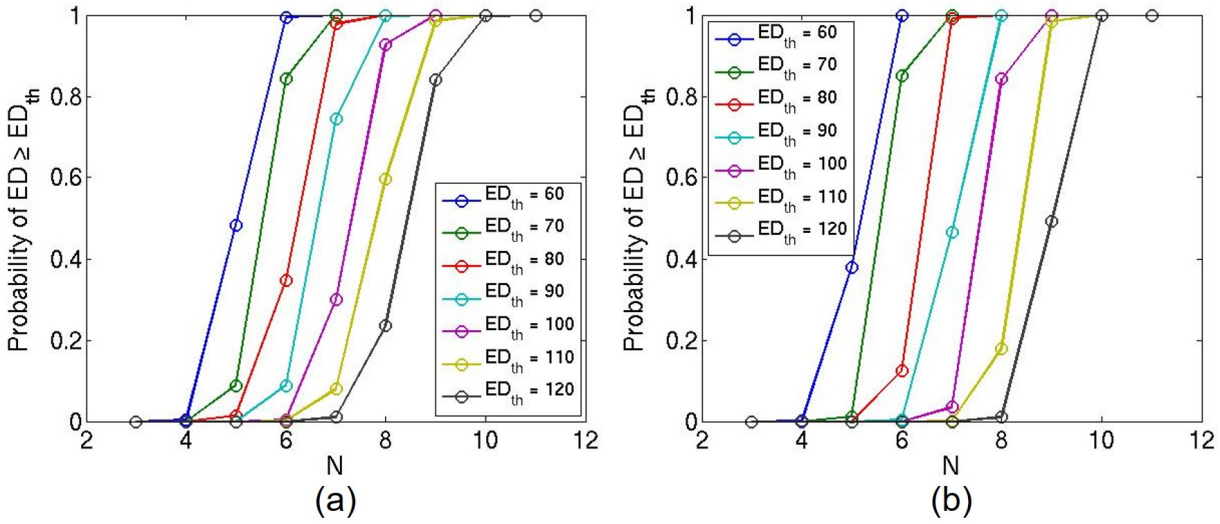


Figure 4.41: Probability of $ED \geq ED_{th}$ increases with N. Shown: $K = 50$ based on (a) Monte Carlo simulation and (b) transition matrix exponentiation.

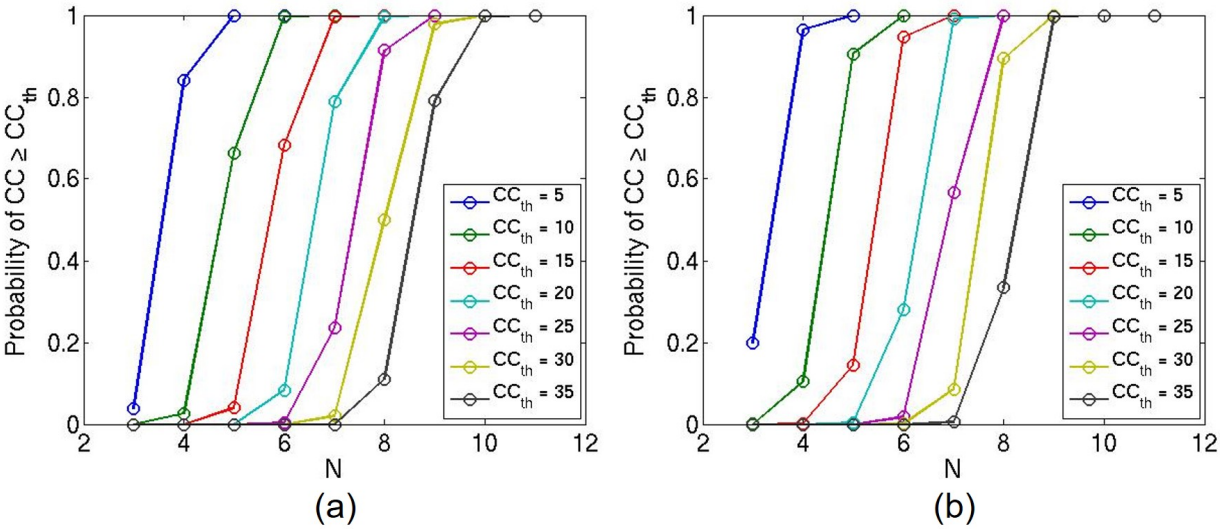


Figure 4.42: Probability of $CC \geq CC_{th}$ increases with N. Shown: $K = 50$ based on (a) Monte Carlo simulation and (b) transition matrix exponentiation.

$CC \geq CC_{th}$ for an $S \times S$ hotspot (ED_{th} and CC_{th} values are normalized to N^2) as follows. For ED, we assume a Gaussian distribution for the distribution of ED after $K \times N^2$ neighbor-swaps. Using the transition matrix exponentiation, we thus find the mean and variance in distance for each lattice edge. We then treat ED for each $S \times S$ window as the sum of Gaussian random variables – the lattice edges composing the window – which gives a Gaussian distribution with mean = $\sum_{latticeedges}\{\text{means}\}$ and variance = $\sum_{latticeedges}\{\text{variances}\}$. We further assume that the window probabilities are independent. Finally, we can approximate the probability that $ED \geq ED_{th}$. For CC, we assume that the probabilities that each of the lattice edges is mapped to an edge that crosses a given fixed window are independent, i.e., each mapped edge's respective crossing of the given window is an independent Bernoulli-distributed random variable. Accordingly, the distribution for the number of crossers for a fixed window (i.e., the number of edges crossing over a fixed window) can be represented as a sum of independent Bernoulli random variables, which is the Poisson Binomial distribution [42]. This is then approximated using the Poisson distribution, with error bound of $9 * \max_{i \in windows} p_i$, as described in [63].

Figure 4.41 and Figure 4.42 respectively show the probabilities of $ED \geq ED_{th}$ and $CC \geq CC_{th}$ for $S = 2$ in an $N \times N$ mesh-like placement with $(K = 50) \cdot N^2$ neighbor-swaps, based on both Monte Carlo simulations (with 500 trials) and transition matrix-based estimations.[68] We observe that the probabilities of having some window's ED or CC value larger than given thresholds increase with N (where $N \times N$ indicates the design size). This observation can also be confirmed by application of the Pigeonhole Principle. Suppose we have $N_1 < N_2$ with corresponding neighbor-swaps $J_1 = K \cdot N_1^2$ and $J_2 = K \cdot N_2^2$, respectively (J_1 and J_2 are absolute values of numbers of neighbor-swaps, and K is a normalized number of neighbor-swaps). By the Pigeonhole Principle, we can find an $N_1 \times N_1$ window within the $N_2 \times N_2$ mesh-like placement that has at least J_1 neighbor-swaps performed on it. Thus, for each window of this $N_1 \times N_1$ design, the probability of exceeding both ED_{th} and CC_{th} will be at least as large as that of the $N_1 \times N_1$ mesh-like placement. This observation indicates that with the same placement quality (i.e., normalized K value with respect to N^2), a larger design is more vulnerable to routing failures.

We also perform a similar study on mesh-like placements. Specifically, we perform some normal-

[68] The small discrepancy between the results from Monte Carlo simulations versus those from transition matrix-based estimations is due to errors of the approximation from the Normal and Poisson distributions. Note that in this section, we use mesh placements, which are more general and can be modeled using the Markov transition matrix technique, instead of real placements.

ized number of neighbor-swaps (K) with respect to the design size (i.e., N^2), and sweep the value of N to study how routing failure probability changes according to N. For each pair (N, K), we perform 10 trials of perturbation and routing, and then report the probability of routing failures. Figure 4.43 shows our results, which empirically support the above analysis that the probability of routing failure increases with the design size, when placement quality as captured by normalized K is kept constant.

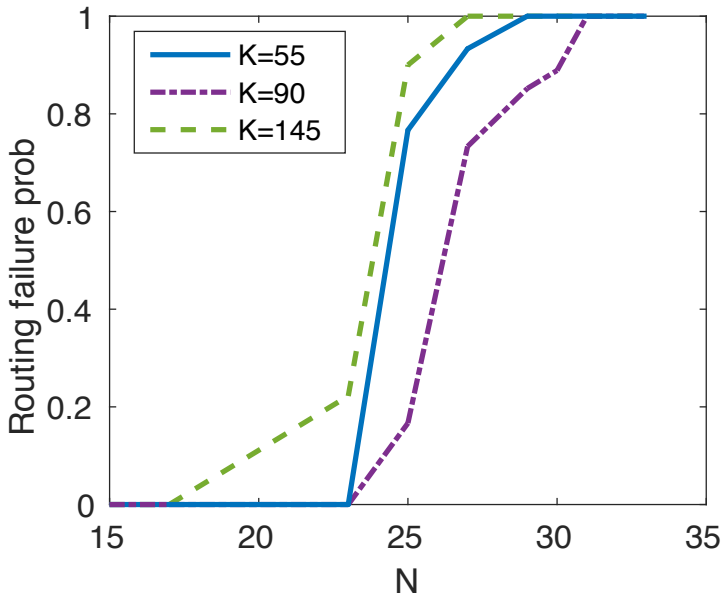


Figure 4.43: Routing failure probabilities versus N for $K = 55$, 90 and 145.

4.3.5 Conclusion

In this work, we propose a systematic framework that measures routing capacity of BEOL stack options as well as inherent capability of routers and, potentially, placers. Using our framework, we demonstrate a "quasi-universal" rank-ordering of various BEOL stacks by routing capacities, for a given router. We also study the relationship between routing hotspot size and placement quality. Lastly, we present an analytical study based on exponentiation of a Markov transition matrix to demonstrate how, with the same placement density and quality, a larger design is more likely to experience routing failures, an observation which is supported by empirical data.

This last observation, that the probability of routing failure increases with the design size (i.e., $N \times N$), is particularly intriguing. This may indicate that there might be *an "optimal block size" for*

dividing an SoC into hard macros for a given design enablement, in a similar spirit to how the authors of [146] proposed 50K to 100K gates as an optimal size of place-and-route blocks nearly 20 years ago.[69] For example, Figure 4.44 shows our notional (hypothesized) tradeoff between design area and block size, which is derived based on several simple assumptions. The x-axis shows block size normalized to the entire chip area, and the y-axis shows area penalty induced by decomposition (i.e., loss of global optimization in hierarchical physical design relative to flat physical design) and/or routability. When the block size is too large, the probability of routing failure is high, and we will end up with a lower utilization to avoid routing failure (orange curve). On the other hand, if the block size is too small, we will have to pay for the cost of partitioning, i.e., loss of optimality due to decomposition (blue curve). To derive the area penalty from routing failure, we first derive a linear model for Δ utilization as a function of $K_{th} - K$ from the result in Figure 4.33 (the routing failure is directly related to maximum achievable utilization, as shown in Section 4.3.3). We then assume that K_{th} is inversely proportional to the square root of the normalized block area, and K is the same for all block sizes (same quality of placements). To derive the area penalty from decomposition, we simply assume that the blocks are connected as clique model, and there is a certain amount of area overhead for each cut. The minimum total area penalty point is shown as the local minimum in the gray curve. According to such a simple model we see in Figure 4.44 that there may be a choice of block size for hard macros in hierarchical design that best compromises between routing failure probability and the cost of partitioning.[70]

Open questions for future researchers might include:

- extension of our framework to comprehend realistic design considerations such as timing, power, manufacturing cost, complex design rules, multi-height cells and power delivery network requirements;

- estimation of "equivalent K" (or another metric to assess and/or rank (windows of) placements with respect to routability) in arbitrary real placements;

[69] The landmark paper of [146] arrived at its predicted block size based on entirely different justifications, namely, scaling of interconnect RC delay and noise, and of gate drive and leakage. The envisioned size of P&R blocks in a hierarchical physical design methodology has evolved differently from the prediction of [146].

[70] Here, our study mainly considers routing and corresponding impacts on timing and power. The actual best choice of block size for hard macros would be dependent on many other aspects of designs and design enablements, e.g., standard cell layouts, netlists, and the characteristics of placers and routers.

- an analytical model to predict optimal block size for minimum area penalty (considering both cost of decomposition and routability) with a given performance (and, possibly, design turnaround time) requirement.

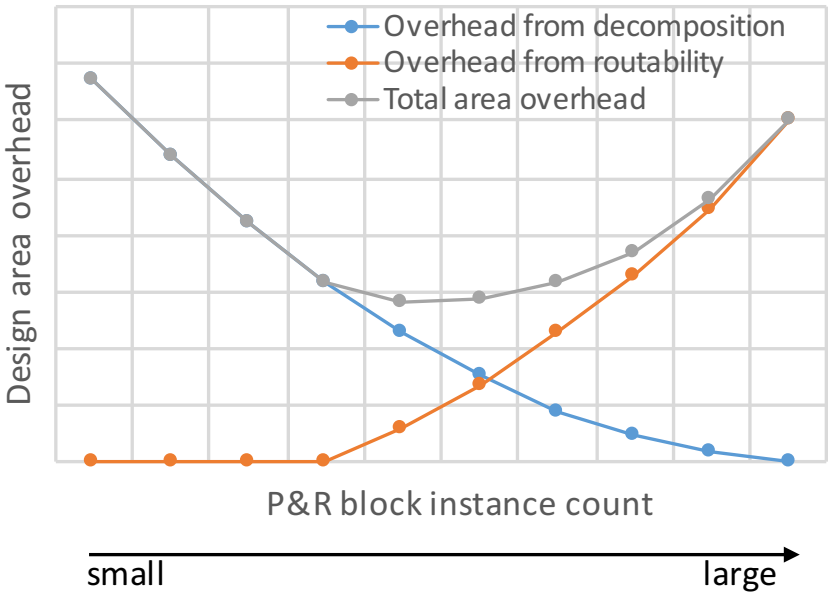


Figure 4.44: A hypothesized tradeoff between overall design area and instance count in individual P&R blocks. The x-axis shows block instance count, and the y-axis shows overall design area overheads induced by (1) decomposition and (2) difficulty of routing.

4.4 Acknowledgments

Chapter 4 contains reprints of Alex Kahng, Andrew B. Kahng, Hyein Lee and Jiajia Li, "PROBE: A Placement, ROuting, Back-End-of-line Measurement Utility", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37(7) 2018; Kwangsoo Han, Andrew B. Kahng, Hyein Lee and Lutong Wang, "Performance- and Energy-Aware Optimization of BEOL Interconnect Stack Geometry in Advanced Technology Nodes", *Proc. International Symposium on Quality Electronic Design*, 2017; and Kwangsoo Han, Andrew B. Kahng and Hyein Lee, "Evaluation of BEOL Design Rule Impacts Using an Optimal ILP-Based Detailed Router", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2015. The dissertation author is a main contributor to, and a primary author of, each of these papers.

I would like to thank my coauthors Kwangsoo Han, Alex Kahng, Andrew B. Kahng, Jiajia Li and Lutong Wang.

Chapter 5

Summary

This thesis has presented new optimization and evaluation methodologies for advanced VLSI manufacturing. The presented methods help overcome new difficulties in physical design arising from advanced design rules as well as technology choices, which are tightly linked to design QoR.

Chapter 2 has presented three distinct gate sizing methodologies that address new challenges in timing signoff and the interaction between sizing and detailed placement. First, we have proposed a new academic sizer, *Sizer*, that reflects a multi-year evolution from a successful "contest" sizing tool to a tool that can outperform high-effort commercial results for a real industrial IC. We describe techniques that are useful in the academic contest setting but not in the real-world context – as well as new techniques specifically developed for Sizer. We also describe the successful application of Sizer to industrial designs. On a design (i.e., *NXPIC*) that is well-optimized by a leading-edge commercial P&R tool as well as by ECO steps, Sizer achieves 7% further leakage reduction without any violation of the given setup/hold and maximum transition constraints in a multi-corner multi-mode context.

Second, we have addressed a new gate sizing/V_t-swapping and placement problem with the *minimum implant area* (MinIA) constraint. The MinIA constraint presents a new challenge to the physical implementation flow in sub-$22nm$ technology, and requires true co-optimization of placement and gate sizing/V_t-swapping. We have proposed sizing and placement heuristics that optimize power and fix MinIA violations while minimizing placement perturbation. Compared to commercial P&R tools, our methods

achieve significant reductions in the number of MinIA violations under timing/power constraints.

Third, we have studied heuristic methods for exploitation of fine-grained mixed-V_t in FDSOI implementation. We explore various implementation flows that include an ILP-based approach and a sensitivity-based heuristic optimization. Based on our experimental results, we summarize fundamental inherent difficulties of fine-grained exploitation of mixed V_t (and body biasing) in FDSOI. We observe that outcomes are highly dependent on designs and library options. Thus, we suggest a decision tree to help designers make early decisions regarding FDSOI implementation choices.

Chapter 3 has presented two distinct methodologies for detailed placement optimization for advanced VLSI manufacturing. First, we have proposed a scalable detailed placement legalization flow for complex FEOL constraints arising at the N10 foundry node. These include drain-drain abutment, minimum implant width, and minimum OD jogging rules. Given initial (timing-driven) placements, our $DFPlacer$ fixes 99% of DRVs with 3% increase in wirelength and minimal impact on timing. We feel that our use case of fixing all but a few tens of violations, with a highly parallelizable two-iteration strategy, is a good practical tradeoff between runtime complexity and DRV fixing. Further, the level of DRV fixing achieved by DFPlacer is encouraging, given that our default experimental configuration makes no attempt at "correctness by construction". Using OpenMP, we confirm that our flow is scalable via a distributed optimization strategy. Additionally, we study an area-DRV tradeoff between two types of standard-cell library strategies, namely, with and without dummy poly gates.

Second, we have proposed a vertical M1 routing-aware detailed placement optimization based on mixed-integer linear programing (MILP) for two new cell architectures in sub-10nm nodes, i.e., *ClosedM1* and *OpenM1*. With our optimization, up to 6.4% (resp. 2.2%) total routed wirelength reductions and 14.4% (resp. 4.1%) #via12 reductions are achieved for *ClosedM1*-based (resp. *OpenM1*-based) designs, with no adverse timing impact.

Chapter 4 has presented three distinct evaluation methodologies for design flows and technology enablements. First, we have studied impacts of patterning technology choices and design rules on physical implementation metrics, with respect to *cost-optimal* design rule-correct detailed routing. We describe *OptRouter*, an ILP-based optimal detailed router that correctly handles multi-pin nets and various sub-20nm routing challenges including via restrictions, via shapes, and SADP patterning rules. OptRouter

enables design rule evaluation using "difficult" routing clips (switchboxes) selected according to a pin cost metric. We study Δcost distributions for different design rules, relative to a baseline rule set, RULE1, wherein all layers are LELE and there are no via restrictions. From the results, we observe that the sensitivities of Δcost to design rules and routing options vary with technology. Also, we observe that there is a gap between pin accessibility metrics such as [148] and our switchbox-centric evaluation of routability.

Second, we have investigated the potential impact of design-aware manufacturing (DAM) and manufacturing-aware design (MAD) methodologies to optimize BEOL dimensions in sub-$10nm$ nodes. We study BEOL interconnect stack geometry by exploring the wire aspect ratio (AR) and duty cycle (DC). We perform SPICE-based analyses of timing path delays and correlate these with analyses in the P&R tool, using a single-stage artificial netlist construction. We also perform block-level studies with placed and routed designs. Based on our studies, we find the optimal (AR,DC) combination for a given wire pitch with respect to power and delay; we also show the sensitivities of BEOL stack geometry to circuit parameters and validate our SPICE analyses with real block-level designs. We further perform studies on design-aware manufacturing and manufacturing-aware design to explore the design freedoms and potential benefits of DAM and MAD. Large differences in design metrics exist across DAM and MAD. By proper utilization of DAM and MAD, we can save up to 60% in TNS and 7% in power for a particular LDPC testcase. Furthermore, based on our experiments, we conjecture that an optimal MAD and DAM BEOL stack exists for any given design.

Third, we have proposed a systematic framework that measures routing capacity of BEOL stack options as well as inherent capability of routers and, potentially, placers. Using our framework, we demonstrate a "quasi-universal" rank-ordering of various BEOL stacks by routing capacities, for a given router. We also study the relationship between routing hotspot size and placement quality. Lastly, we present an analytical study based on exponentiation of a Markov transition matrix to demonstrate how, with the same placement density and quality, a larger design is more likely to experience routing failures, an observation which is supported by empirical data.

Bibliography

[1] S. N. Adya, M. C. Yildiz, I. L. Markov, P. G. Villarrubia, P. N. Parakh and P. H. Madden, "Benchmarking for Large-Scale Placement and Beyond", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 23(4) (2004), pp. 472-487.

[2] R. Aitken, *personal communication*, March 2015.

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

[4] D. Albano, M. Lanuzza, R. Taco and F. Crupi, "Gate-Level Body Biasing for Subthreshold Logic Circuits: Analytical Modeling and Design Guidelines", *International Journal of Circuit Theory and Applications* 43(11) (2015), pp. 1523-1540.

[5] D. Aldous and U. Vazirani, "'Go with the winners' algorithms", *Proc. Symposium on Foundations of Computer Science*, 1994, pp. 492-501.

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

[7] C. J. Alpert, G. J. Nam and P. G. Villarrubia, "Effective Free Space Management for Cut-Based Placement via Analytical Constraint Generation", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 22(10) (2003), pp. 1343-1353.

[8] M. B. Anand, H. Shibata and M. Kakumu, "Optimization Study of VLSI Interconnect Parameters", *IEEE Transactions on Electron Devices* 47(1) (2000), pp. 178-186.

[9] M. B. Anand, H. Shibata and M. Kakumu, "Multiobjective Optimization of VLSI Interconnect Parameters", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 17(12) (1998), pp. 1252-1261.

[10] S. R. Asmussen, *Applied Probability and Queues*, New York, Springer, 2003.

[11] Y. Badr, K.-W. Ma and P. Gupta, "Layout Pattern-Driven Design Rule Evaluation", *Proc. SPIE, Design-Process-Technology Co-optimization for Manufacturability VIII*, 2014, pp. 043018-1-043018-8

[12] H. B. Bakoglu and J. D. Meindl, "Optimal Interconnection Circuits for VLSI", *IEEE Transactions on Electron Devices* 32(5) (1985), pp. 903-909.

[13] M. R. C. M. Berkelaar and J. A. G. Jess, "Gate Sizing in MOS Digital Circuits with Linear Programming", *Proc. European Design Automation Conference*, 1990, pp. 217-221.

[14] S. Block, W. Daub, J. Dirks, F. Labib, R. Mann, H. Narisetty, H. Preuthen, F. Pugliese, T. Ragheb and R. Trihy, "Entering FD-SOI Era Using GLOBALFOUNDRIES 22FDX Technology", 2016. https://www.globalfoundries.com/sites/default/files/articles/entering-fd-soi-era-using-globalfoundries-22fdx-technology.pdf

[15] U. Brenner and A. Rohe, "An Effective Congestion Driven Placement Framework", *Proc. ACM International Symposium on Physical Design*, 2002, pp. 6-11.

[16] U. Brenner and J. Vygen, "Faster Optimal Single-Row Placement with Fixed Ordering", *Proc. Design, Automation and Test in Europe*, 2000, pp. 117-121.

[17] F. Brglez, D. Bryan and K. Koźmiński, "Combinational Profile of Sequential Benchmark Circuits", *Proc. IEEE International Symposium on Circuits and Systems*, 1989, pp. 1929-1933.

[18] F. Brglez and H. Fujiwara, "Recent Algorithms for Gate-Level ATPG with Fault Simulation and Their Performance Assessment", *Proc. IEEE International Symposium on Circuits and Systems*, 1985, pp. 663-698.

[19] A. E. Caldwell, A. B. Kahng, A. A. Kennings and I. L. Markov, "Hypergraph Partitioning for VLSI CAD: Methodology for Heuristic Development, Experimentation and Reporting", *Proc. ACM/IEEE Design Automation Conference*, 1999, pp. 349-354.

[20] R. Carden and C.K. Cheng, "A Global Router with a Theoretical Bound on the Optimal Solution", *IEEE Trans. on CAD* 15(2) (1996), pp. 208-216.

[21] A. Chakraborty, S. X. Shi and D. Z. Pan, "Stress Aware Layout Optimization Leveraging Active Area Dependent Mobility Enhancement", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 29(10) (2010), pp. 1533-1545.

[22] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze and M. Xie, "mPL6: Enhanced Multilevel Mixed-Size Placement", *Proc. ACM International Symposium on Physical Design*, 2006, pp. 212-214.

[23] T. B. Chan, A. B. Kahng and J. Li, "Toward Quantifying the IC Design Value of Interconnect Technology Improvements", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2013, pp. 1-6.

[24] W.-T. J. Chan, Y. Du, A. B. Kahng, S. Nath and K. Samadi, "BEOL Stack-Aware Routability Prediction from Placement Using Data Mining Techniques", *Proc. IEEE International Conference on Computer Design*, 2016, pp. 41-48.

[25] C. C. Chang, J. Cong and M. Xie, "Optimality and Scalability Study of Existing Placement Algorithms", *Proc. Asia and South Pacific Design Automation Conference*, 2003, pp. 621-627.

[26] C.-P. Chen, C. Chi and D. F. Wong, "Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 18(7) (1999), pp. 1014-1025.

[27] T.-C. Chen, T.-C. Hsu, Z.-W. Jiang and Y.-W. Chang, "NTUplace: A Ratio Partitioning Based Placement Algorithm for Large-Scale Mixed-Size Designs", *Proc. ACM International Symposium on Physical Design*, 2005, pp. 236-238.

[28] M. Cho and D. Z. Pan, "BoxRouter: A New Global Router Based on Box Expansion and Progressive ILP", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26(12) (2007), pp. 2130-2143.

[29] H. Chou, Y.-H. Wang, and C. C.-P. Chen, "Fast and Effective Gate Sizing with Multiple-V_t Assignment Using Generalized Lagrangian Relaxation", *Proc. Asia and South Pacific Design Automation Conference*, 2005, pp. 381-386.

[30] I. Ciofi, A. Contino, P. J. Roussel, R. Baert, V. H. Vega-Gonzalez, K. Croes, M. Badaroglu, C. J. Wilson, P. Raghavan, A. Mercha and D. Verkest, "Impact of Wire Geometry on Interconnect RC and Circuit Delay", *IEEE Trans. on Electron Devices* 63(6) (2016), pp. 2488-2496.

[31] J. Cong, M. Romesis and M. Xie, "Optimality, Scalability and Stability Study of Partitioning and Placement Algorithms", *Proc. ACM International Symposium on Physical Design*, 2003, pp. 88-94.

[32] P. Corsonello, M. Lanuzza and S. Perri, "Gate-Level Body Biasing Technique for High Speed Sub-Threshold CMOS Logic Gates", *International Journal of Circuit Theory and Applications* 42(1) (2014), pp. 65-70.

[33] R. L. Ching, E. F. Young, K. C. Leung and C. Chu, "Post-Placement Voltage Island Generation", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2006, pp. 641-646.

[34] D. G. Connery and K. Keutzer, "Linear Programming for Sizing, V_{th} and V_{dd} Assignment", *Proc. International Symposium on Low Power Electronic Design*, 2005, pp. 149-154.

[35] F. Corno, M. S. Reorda and G. Squillero, "RT-Level ITC'99 Benchmarks and First ATPG Results", *IEEE Design & Test of Computers* 17(3) (2000), pp. 44-53.

[36] J. Darnauer and W. W.-M. Dai, "A Method for Generating Random Circuits and Its Application to Routability Measurement", *Proc. ACM International Symposium on Field-Programmable Gate Arrays*, 1996, pp. 66-72.

[37] S. Dobre, A. B. Kahng and J. Li, "Design Implementation with Non-Integer Multiple-Height Cells for Improved Design Quality in Advanced Nodes", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37(4) (2018), pp. 855-868.

[38] X. Dong, J. Zhao and Y. Xie, "Fabrication Cost Analysis and Cost-Aware Design Space Exploration for 3D-ICs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 29(12) (2010), pp. 1959-1972.

[39] Y. Du and M. D. F. Wong, "Optimization of Standard Cell Based Detailed Placement for 16nm FinFET Process", *Proc. Design, Automation and Test in Europe*, 2014, pp. 1-6.

[40] I. M. Elfadel, M. B. Anand, A. Deutsch, O. Adekanmbi, M. Angyal, H. Smith, B. Rubin and G. Kopcsay, "AQUAIA: A CAD Tool for On-Chip Interconnect Modeling, Analysis, and Optimization", *Proc. Electrical Performance of Electronic Packaging*, 2002, pp. 337-340.

[41] M. G. Faruk, M. S. Angyal, O. Ogunsola, D. K. Watts and R. Wilkins, "Variability Modeling and Process Optimization for the 32 nm BEOL Using In-Line Scatterometry Data", *IEEE Trans. on Semiconductor Manufacturing* 27(2) (2014), pp. 260-268.

[42] M. Fernandez and S. Williams, "Closed-Form Expression for the Poisson-Binomial Probability Density Function", *IEEE Transactions on Aerospace and Electronic Systems* 46(2) (2010), pp. 803-817.

[43] J. P. Fishburn and A. E. Dunlop, "Tilos: A Posynomial Programming Approach to Transistor Sizing", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 1985, pp. 326-328.

[44] G. Flach, T. Reimann, G. Posser, M. Johann and R. Reis, "Effective Method for Simultaneous Gate Sizing and Vth Assignment Using Lagrangian Relaxation", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33(4) (2014), pp. 546-557.

[45] P. Flatresse, "UTBB-FDSOI Design & Migration Methodology", 2013. https://mycmp.fr/IMG/pdf/utbb-fdsoidesign_migration_methodology_.pdf

[46] A. C. Flores, "Layout Floorplan using Body Bias Islands", *M.S. thesis*, TU Eindhoven, August, 2010.

[47] R. S. Ghaida, Y. Badr, M. Gupta, N. Jin and P. Gupta, "Comprehensive Die-Level Assessment of Design Rules and Layouts", *Proc. Asia and South Pacific Design Automation Conference*, 2014, pp. 61-66.

[48] R. S. Ghaida and P. Gupta, "DRE: A Framework for Early Co-Evaluation of Design Rules, Technology Choices, and Layout Methodologies", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31(9) (2012), pp. 1379-1392.

[49] F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, 1999.

[50] P. Grassberger, "Go with the Winners: A General Monte Carlo Strategy", *http://arxiv.org/pdf/cond-mat/0201313v1.pdf*.

[51] L. Guo, Y. Cai, Q. Zhou and X. Hong, "Logic and Layout Aware Voltage Island Generation for Low Power Design", *Proc. Asia and South Pacific Design Automation Conference*, 2007, pp. 666-671.

[52] M. Gupta, K. Jeong and A. B. Kahng, "Timing Yield-Aware Color Reassignment and Detailed Placement Perturbation for Bimodal CD Distribution in Double Patterning Lithography", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 29(8) (2010), pp. 1129-1242.

[53] P. Gupta, K. Jeong, A. B. Kahng and C.-H. Park, "Electrical Assessment of Lithographic Gate Line-End Patterning", *SPIE Journal of Microlithography, Microfabrication and Microsystems* 9(2) (2010), pp. 023014-1-023014-19.

[54] P. Gupta and A. B. Kahng, "Manufacturing-Aware Physical Design", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2003, pp. 681-687.

[55] P. Gupta, A. B. Kahng, A. Kasibhatla and P. Sharma, "Eyecharts: Constructive Benchmarking of Gate Sizing Heuristics", *Proc. ACM/IEEE Design Automation Conference*, 2010, pp. 592-602.

[56] P. Gupta, A. B. Kahng and C.-H. Park, "Detailed Placement for Improved Depth of Focus and CD Control", *Proc. Asia and South Pacific Design Automation Conference*, 2005, pp. 343-348.

[57] P. Gupta, A. B. Kahng and C.-H. Park, "Manufacturing-Aware Design Methodology for Assist Feature Correctness", *Proc. SPIE*, 2005, pp. 131-140.

[58] P. Gupta, A. B. Kahng, P. Sharma and D. Sylvester, "Gate-Length Biasing for Runtime-Leakage Control", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25(8) (2006), pp. 1475-1485.

[59] L. W. Hagen, D. J.-H. Huang and A. B. Kahng, "Quantified Suboptimality of VLSI Layout Heuristics", *Proc. ACM/IEEE Design Automation Conference*, 1995, pp. 216-221.

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

[61] K. Han, A. B. Kahng and H. Lee, "Scalable Detailed Placement Legalization for Complex Sub-14nm Constraints", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2015, pp. 867-873.

[62] 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", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2013, pp. 1-6.

[63] J. L. Hodges and L. L. Cam, "The Poisson Approximation to the Poisson Binomial Distribution", *Ann. Math. Statist.* 31(3) (1960), pp. 737-740.

[64] T. C. Hu, A. B. Kahng and C. W. Tsao, "Old Bachelor Acceptance: A New Class of Non-Monotone Threshold Accepting Methods", *ORSA J. on Computing* 7(4) (1995), pp. 417-425.

[65] S. Hu, M. Ketkar and J. Hu, "Gate Sizing for Cell-Library-Based Designs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28(6) (2009), pp. 818-825.

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

[67] J. Hu, J. A. Roy and I. L. Markov, "Sidewinder: A Scalable ILP-based Router", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2008, pp. 73-80.

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

[69] Y. L. Huang, J. Hu and W. Shi, "Lagrangian Relaxation for Gate Implementation Selection", *Proc. ACM International Symposium on Physical Design*, 2011, pp. 167-174.

[70] S.-W. Hur and J. Lillis, "Mongrel: Hybrid Techniques for Standard Cell Placement", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2000, pp. 165-170.

[71] M. D. Hutton, J. Rose, J. P. Grossman and D. Corneil, "Characterization and Parameterized Generation of Synthetic Combinational Circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 17(10) (1998), pp. 985-996.

[72] IBM (Physical synthesis optimization research staff member), *personal communication*, October 2015.

[73] C. Inacio, H. Schmit, D. Nagle, A. Ryan, D. E. Thomas, Y. Tong and B. Klass, "Vertical Benchmarks for CAD", *Proc. ACM/IEEE Design Automation Conference*, 1999, pp. 408-413.

[74] Y. I. Ismail, E. G. Friedman and J. L. Neves, "Equivalent Elmore Delay for RLC Trees", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 19(1) (2000), pp. 83-97.

[75] D. Jacquet et al., "A 3 GHz Dual Core Processor ARM Cortex TM -A9 in 28 nm UTBB FD-SOI CMOS With Ultra-Wide Voltage Range and Energy Efficiency Optimization", *IEEE Journal of Solid State Circuits* 49(4) (2014) pp. 812-826.

[76] C.-H. Jan, B. Uddalak, R. Brain, S.-J. Choi, G. Curello, G. Gupta and W. Hafez, "A 22nm SoC Platform Technology Featuring 3-D Tri-gate and High-k/Metal Gate, Optimized for Ultra Low Power, High Performance and High Density SoC Applications", *Proc. IEEE International Electron Devices Meeting*, 2012, pp. 311-314.

[77] K. Jeong, A. B. Kahng and H. Yao, "Revisiting the Linear Programming Framework for Leakage Power vs. Performance Optimization", *Proc. International Symposium on Quality Electronic Design*, 2009, pp. 127-134.

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

[79] Q.-T. Jiang, M.-H. Tsai and R. H. Havemann, "Line Width Dependence of Copper Resistivity", *Proc. Interconnect Technology Conference*, 2001, pp. 227-229.

[80] V. Joshi, B. Cline, D. Sylvester, D. Blaauw and K. Agarwal, "Leakage Power Reduction Using Stress-Enhanced Layouts", *Proc. ACM/IEEE Design Automation Conference*, 2008, pp. 912-917.

[81] A. B. Kahng, "The ITRS Design Technology and System Drivers Roadmap: Process and Status", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2013, pp. 34-39.

[82] A. B. Kahng, "New Game, New Goal Posts: A Recent History of Timing Closure", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2015, pp. 1-6.

[83] A. B. Kahng and S. Kang, "Construction of Realistic Gate Sizing Benchmarks With Known Optimal Solutions", *Proc. ACM International Symposium on Physical Design*, 2012, pp. 153-160.

[84] A. B. Kahng, S. Kang, H. Lee, I. L. Markov and P. Thapar, "High-Performance Gate Sizing with a Signoff Timer", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2013, pp. 450-457.

[85] A. B. Kahng, S. Kang, H. Lee, S. Nath and J. Wadhwani, "Learning-Based Approximation of Interconnect Delay and Slew in Signoff Timing Tools", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2013, pp. 1-8.

[86] A. B. Kahng and H. Lee, "Minimum Implant Area-Aware Gate Sizing and Placement", *Proc. ACM Great Lakes Symposium on Very Large Scale Integration*, 2014, pp. 57-62.

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

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

[89] A. B. Kahng, I. L. Markov and S. Reda, "On Legalization of Row-Based Placements", *Proc. ACM Great Lakes Symposium on Very Large Scale Integration*, 2004, pp. 214-219.

[90] A. B. Kahng and S. Reda, "Evaluation of Placer Suboptimality Via Zero-Change Netlist Transformations", *Proc. ACM International Symposium on Physical Design*, 2005, pp. 208-215.

[91] A. B. Kahng, P. Sharma and R. O. Topaloglu, "Exploiting STI Stress for Performance", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2007, pp. 83-90.

[92] A. B. Kahng, P. Tucker and A. Zelikovsky, "Optimization of Linear Placements for Wirelength Minimization with Free Sites", *Proc. Asia and South Pacific Design Automation Conference*, 1999, pp. 241-244.

[93] A. B. Kahng and X. Xu, "Accurate Pseudo-Constructive Wirelength and Congestion Estimation", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2003, pp. 61-68.

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

[95] J. M. Kühn, H. Amano, W. Rosenstiel and O. Bringmann, "Leveraging FDSOI through body bias domain partitioning and bias search", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2016, pp. 1-6.

[96] B. S. Landman and R. L. Russo, "On a Pin Versus Block Relationship For Partitions of Logic Graphs", *IEEE Transactions on Computers* C-20(12) (1971), pp. 1469-1479.

[97] J. Lee and P. Gupta, "Incremental Gate Sizing for Late Process Changes", *Proc. IEEE International Conference on Computer Design*, 2010, pp. 215-221.

[98] J. Lee and P. Gupta, "Discrete Circuit Optimization", *Foundations and Trends in Electronic Design Automation* 6(1) (2012), pp. 1-120.

[99] L. Li, P. Kang, Y. Lu and H. Zhou, "An Efficient Algorithm for Library-based Cell-type Selection in High-Performance Low-Power Designs", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2012, pp. 226-232.

[100] S. Li and C.-K. Koh, "Mixed Integer Programming Models for Detailed Placement", *Proc. ACM International Symposium on Physical Design*, 2012, pp. 87-94.

[101] S. Li and C.-K. Koh, "MIP-based Detailed Placer for Mixed-size Circuits", *Proc. ACM International Symposium on Physical Design*, 2014, pp. 11-18.

[102] J. Li, B. Yang, X. Hu, Q. Dong and S. Nakatake, "STI Stress Aware Placement Optimization Based On Geometric Programming", *Proc. ACM Great Lakes Symposium on Very Large Scale Integration*, 2009, pp. 209-214.

[103] L. Liebmann, V. Gerousis, P. Gutwin, M. Zhang, G. Han and B. Cline, "Demonstrating Production Quality Multiple Exposure Patterning Aware Routing for the 10NM Node", *Proc. SPIE, Design-Process-Technology Co-optimization for Manufacturability VIII*, 2014, pp. 905309-1-905309-10.

[104] L. Liebmann and D. Pietromonaco, "The Increasing Pain of Scaling with 193i: Where Does it Hurt? How Much More Can We Endure?", *keynote*, *SPIE Optical Microlithography*, 2013.

[105] T. Lin and C. Chu, "TPL-Aware Displacement-driven Detailed Placement Refinement with Coloring Constraints", *Proc. ACM International Symposium on Physical Design*, 2015, pp. 75-80.

[106] Y. Lin, B. Yu, B. Xu and D. Z. Pan. "Triple Patterning Aware Detailed Placement Toward Zero Cross-Row Middle-of-Line Conflict", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2015, pp. 396-403.

[107] B. Liu, Y. Cai, Q. Zhou, and X. Hong, "Power Driven Placement with Layout Aware Supply Voltage Assignment for Voltage Island Generation in Dual-Vdd Designs", *Proc. Asia and South Pacific Design Automation Conference*, 2006, pp. 582-587.

[108] Q. Liu and M. Marek-Sadowska, "Pre-layout Wire Length and Congestion Estimation", *Proc. ACM/IEEE Design Automation Conference*, 2004, pp. 582-587.

[109] W.-H. Liu, T.-K. Chien and T.-C. Wang, "Region-Based and Panel-Based Algorithms for Unroutable Placement Recognition", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34(4) (2015), pp. 502-514.

[110] W.-H. Liu, Y. Wei, C. Sze, C. J. Alpert, Z. Li, Y.-L. Li and N. Viswanathan, "Routing Congestion Estimation with Real Design Constraints", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2013, pp. 1-8.

[111] Y. Liu and J. Hu, "A New Algorithm for Simultaneous Gate Sizing and Threshold Voltage Assignment", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 29(2) (2010), pp. 223-234.

[112] V. S. Livertamento, C. Guth, J. L. Güntzel and M. O. Johann, "Fast and Efficient Lagrangian Relaxation-Based Discrete Gate Sizing", *Proc. Design, Automation and Test in Europe*, 2013, pp. 1855-1860.

[113] V. S. Livramento, C. Guth, J. L. Güntzel and M. O. Johann, "A Hybrid Technique for Discrete Gate Sizing Based on Lagrangian Relaxation", *ACM Transactions on Design Automation of Electronic Systems* 19(4) (2014), no. 40.

[114] T. Luo, D. Newmark and D. Z. Pan, "Total Power Optimization Combining Placement, Sizing and Multi-Vt Through Slack Distribution Management", *Proc. Asia and South Pacific Design Automation Conference*, 2008, pp. 352-357.

[115] N. D. MacDonald, "Timing Closure in Deep Submicron Designs", DAC.com Knowledge Center Article, March 2010.
http://vlsicad.ucsd.edu/DAC15/MACDONALD_TIMINGCLOSURE.pdf

[116] O. Martin, S. W. Otto and E. W. Felten, "Large-Step Markov Chains for the Traveling Salesman Problem", *Complex Systems* 5(3) (1991), pp. 299–326.

[117] C. Moon, P. Gupta, P. J. Donehue and A. B. Kahng, "Designing a Digital Circuit by Correlating Different Static Timing Analyzers", *U.S. Patent* No. 7,823,098, 2010.

[118] S. Narasimha, K. Onishi, H. M. Nayfeh, A. Waite, M. Weybright, J. Johnson, C. Fonseca et al., "High Performance 45-nm SOI Technology with Enhanced Strain, Porous Low-k BEOL, and Immersion Lithography", *Proc. IEEE International Electron Devices Meeting*, 2006, pp. 1-4.

[119] M. M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke and C. Zhuo, "ISPD-2012 Discrete Cell Sizing Contest and Benchmark Suite", *Proc. ACM International Symposium on Physical Design*, 2012, pp. 161-164.

[120] M. M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke and C. Zhuo, "An Improved Benchmark Suite for the ISPD-2013 Discrete Cell Sizing Contest", *Proc. ACM International Symposium on Physical Design*, 2013, pp. 168-170.

[121] M. M. Ozdal, S. Burns and J. Hu, "Gate Sizing and Device Technology Selection Algorithms for High-Performance Industrial Designs", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2011, pp. 724-731.

[122] M. M. Ozdal, S. Burns and J. Hu, "Algorithm for Gate Sizing and Device Parameter Selection for High-Performance Designs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31(10) (2012), pp. 1558-1571.

[123] D. Pamunuwa and H. Tenhunen, "Repeater Insertion to Minimise Delay in Coupled Interconnects", *Proc. IEEE/ACM International Conference on VLSI Design*, 2001, pp. 513-517.

[124] M. Pan and C. Chu, "IPR: An Integrated Placement and Routing Algorithm", *Proc. ACM/IEEE Design Automation Conference*, 2007, pp. 59-62.

[125] M. Pan and C. Chu, "FastRoute 2.0: A High-quality and Efficient Global Router", *Proc. Asia and South Pacific Design Automation Conference*, 2007, pp. 250-255.

[126] Z. Qi, Y. Cai and Q. Zhou, "Accurate Prediction of Detailed Routing Congestion Using Supervised Data Learning", *Proc. IEEE International Conference on Computer Design*, 2014, pp. 97-103.

[127] Qualcomm Technologies Inc. VLSI technology principal engineer, *personal communication*, July 2014.

[128] Qualcomm Technologies Inc., Physical synthesis optimization senior staff engineer, *personal communication*, November 2018.

[129] M. Rahman and C. Sechen, "Post-synthesis Leakage Power Minimization", *Proc. Design, Automation and Test in Europe*, 2012, pp. 99-104.

[130] M. Rahman, H. Tennakoon and C. Sechen, "Power Reduction via Near-Optimal Library-Based Cell-Size Selection", *Proc. Design, Automation and Test in Europe*, 2011, pp. 867-870.

[131] M. Rahman, H. Tennakoon and C. Sechen, "Library-Based Cell-Size Selection Using Extended Logical Effort", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32(7) (2013), pp. 1086-1099.

[132] P. Ramachandaran, A. R. Agnihotri, S. Ono, P. Damodaran, K. Srihari and P. H. Madden, "Optimal Placement by Branch-and-Price", *Proc. Asia and South Pacific Design Automation Conference*, 2005, pp. 337-342.

[133] T. Reimann, G. Posser, G. Flach, M. Johann and R. Reis, "Simultaneous Gate Sizing and Vt Assignment using Fanin/Fanout Ratio and Simulated Annealing", *Proc. IEEE International Symposium on Circuits and Systems*, 2013, pp. 2549-2552.

[134] T. Reimann, C. C. N. Sze and R. Reis, "Gate Sizing and Threshold Voltage Assignment for High Performance Microprocessor Designs", *Proc. Asia and South Pacific Design Automation Conference*, 2015, pp. 214-219.

[135] L. Remy, P. Coll, F. Picot, P. Mico and J.-M. Portal, "Definition of an Innovative Filling Structure for Digital Blocks: the DFM Filler Cell", *Proc. IEEE International Conference on Electronics, Circuits and Systems*, 2009, pp. 73-76.

[136] S. Roy, D. Liu, J. Um and D. Z. Pan, "OSFA: A New Paradigm of Gate-sizing for Power/Performance Optimizations under Multiple Operating Conditions", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2015, pp. 1-6.

[137] J. A. Roy and I. L. Markov, "High-performance Routing at the Nanometer Scale", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2007, pp. 496-502.

[138] M. Saeedi, M . S. Zamani and A. Jahanian, "Prediction and Reduction of Routing Congestion", *Proc. ACM International Symposium on Physical Design*, 2006, pp. 72-77.

[139] Samsung Electronics Corp. (CAE principal engineer), *personal communication*, September 2014.

[140] P. P. Shah, "Optimization of the BEOL Interconnect Stack for Advanced Semiconductor Technology Nodes", *M.S. Thesis*, UC San Diego ECE Department, 2015.

[141] B. Sluijk et al., ASML, *personal communication*, December 2015.

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

[143] L. Srivani and V. Kamakoti, "Synthetic Benchmark Digital Circuits: A Survey", *IETE Technical Review* 29(6) (2012), pp. 442-448.

[144] A. Srivastava, D. Sylvester and D. Blaauw, "Power Minimization Using Simultaneous Gate Sizing, Dual-V_{dd} and Dual-V_{th} Assignment", *Proc. ACM/IEEE Design Automation Conference*, 2004, pp. 783-787.

[145] D. Stroobandt, P. Verplaetse and J. Van Campenhout, "Generating Synthetic Benchmark Circuits for Evaluating CAD Tools", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 19(9) (2000), pp. 1011-1022.

[146] D. Sylvester and K. Keutzer, "Getting to the Bottom of Deep Submicron", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 1998, pp. 203-211.

[147] R. Taco, I. Levi, M. Lanuzza and A. Fish, "Low Voltage Logic Circuits Exploiting Gate Level Dynamic Body Biasing in 28nm UTBB FD-SOI", *Solid-State Electronics* 117 (2016), pp. 185-192.

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

[149] S. Takahashi, M. Edahiro and Y. Hayashi, "Interconnect Design Strategy: Structures, Repeaters and Materials with Strategic System Performance Analysis Model", *IEEE Trans. on Electron Devices* 48(2) (2001), pp. 239-251.

[150] H. Tennakoon and C. Sechen, "Gate Sizing Using Lagrangian Relaxation Combined with a Fast Gradient-Based Pre-Processing Step", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2002, pp. 395-402.

[151] B. Tutuianu, F. Dartu and L. Pileggi, "An Explicit RC-Circuit Delay Approximation Based on the First Three Moments of the Impulse Response", *Proc. ACM/IEEE Design Automation Conference*, 1996, pp. 611-616.

[152] N. Viswanathan, C. J. Alpert, C. Sze, Z. Li, G.-J. Nam and J. A. Roy, "The ISPD-2011 Routability-Driven Placement Contest and Benchmark Suite", *Proc. ACM International Symposium on Physical Design*, 2011, pp. 141-146.

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

[154] L. Wei, K. Roy and C. Koh, "Power Minimization by Simultaneous Dual-V_{th} Assignment and Gate-Sizing", *Proc. IEEE Custom Integrated Circuits Conference*, 2000, pp. 413-416.

[155] Y. Wei, C. Sze, N. Viswanathan, Z. Li, C. J. Alpert, L. Reddy, A. D. Huber, G. E. Tellez, D. Keller and S. S. Sapatnekar, "GLARE: Global and Local Wiring Aware Routability Evaluation", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2012, pp. 768-773.

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

[157] T.-H. Wu and A. Davoodi, "PaRS: Parallel and Near-Optimal Grid-Based Cell Sizing for Library-Based Design", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28(11) (2009), pp. 1666-1678.

[158] H. Wu, M. D. F. Wong, I.-M. Liu and Y. Wang, "Placement-Proximity-Based Voltage Island Grouping Under Performance Requirement", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26(7) (2007), pp. 1256-1269.

[159] H. Xiang, H. Qian, C. Zhou, Y.-S. Lin, F. Yee, A. Sullivan and P.-F. Lu, "Row Based Dual-VDD Island Generation and Placement", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2014, pp. 1-6.

[160] H. Xiang, L. Reddy, H. Qian, C. Zhou, Y.-S. Lin, F. Yee, A. Sullivan and P.-F. Lu, "Gate Movement for Timing Improvement on Row Based Dual-VDD Designs", *Proc. International Symposium on Quality Electronic Design*, 2016, pp. 423-429.

[161] J. Xie and C.Y. R. Chen, "Lookup Table Based Discrete Gate Sizing for Delay Minimization with Modified Elmore Delay Model", *Proc. ACM Great Lakes Symposium on Very Large Scale Integration*, 2015, pp. 361-366.

[162] X. Xu, B. Cline, G. Yeric, B. Yu and D. Z. Pan, "Self-Aligned Double Patterning Aware Pin Access and Standard Cell Layout Co-Optimization", *Proc. ACM International Symposium on Physical Design*, 2014, pp. 101-108.

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

[164] B. Yu, X. Xu, J.-R. Gao and D. Z. Pan, "Methodology for Standard Cell Compliance and Detailed Placement for Triple Patterning Lithography", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2013, pp. 349-356.

[165] C. Yeh, Y.-S. Kang, S.-J. Shieh and J.-S. Wang, "Layout Techniques Supporting the Use of Dual Supply Voltages for Cell-Based Designs", *Proc. ACM/IEEE Design Automation Conference*, 1999, pp. 62-67.

[166] Q. Zhou, X. Wang, Z. Qi, Z. Chen, Q. Zhou and Y. Cai, "An Accurate Detailed Routing Routability Prediction Model in Placement", *Proc. Asia Symposium on Quality Electronic Design*, 2015, pp. 119-122.

[167] J. J. Zhu, P. Chidambaram, G. Nallapati and C. F. Yeap, "Back End of Line (BEOL) Local Optimization to Improve Product Performance", *U.S. Patent Application 20150303145 A1*, 2014.

[168] Z. Zhu, D. Wan and Y. Yang, "An Interconnect-Line-Size Optimization Model Considering Scattering Effect", *IEEE Electron Device Letters* 31(7) (2010), pp. 641-643.

[169] ASAP ASU 7nm PDK. http://asap.asu.edu/asap/

[170] Blaze MO. http://www.tela-inc.com

[171] Cadence Encounter Digital Implementation System User Guide, http://www.cadence.com

[172] Cadence Encounter Timing System User Guide, http://www.cadence.com

[173] *Cadence Genus Synthesis Solution User's Manual*, http://www.cadence.com

[174] *Cadence Innovus User's Manual*, http://www.cadence.com

[175] Cadence Quantus QRC User Guide, http://www.cadence.com

[176] Dorado. http://www.dorado-da.com/

[177] Gary Smith EDA. http://www.garysmitheda.com/

[178] *Global Foundries: 22FDX*, https://www.globalfoundries.com/technology-solutions/cmos/fdx/22fdx

[179] Horizontal Benchmarking Extension Project Website. http://vlsicad.ucsd.edu/A2A

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

[181] ISPD05/06 benchmarks.
http://archive.sigda.org/ispd2005/contest.htm, http://archive.sigda.org/ispd2006/contest.html

[182] International Technology Roadmap for Semiconductors. http://www.itrs2.net

[183] ITRS 2011 Design Chapter. http://www.itrs2.net/2011-itrs.html

[184] ITRS Low-Power Design Technology Roadmap, Design Chapter Table DESN14, 2011.
http://www.itrs2.net/2011-itrs.html

[185] ITRS Lithography Chapter, 2013. http://www.itrs2.net

[186] LEF DEF reference. http://www.si2.org/openeda.si2.org/projects/lefdef

[187] Liberty Technical Advisory Board. http://www.opensourceliberty.org

[188] Mentor Graphics Calibre, https://www.mentor.com

[189] Mentor Graphics Olympus-SoC, https://www.mentor.com

[190] NXP Semiconductors. http://www.nxp.com

[191] OpenCores: Open Source IP-Cores. http://www.opencores.org

[192] OpenMP Architecture Review Board, “OpenMP Application Program Interface, Version 3.1”.
https://www.openmp.org

[193] Si2 OpenAccess. http://www.si2.org/openaccess

[194] Si2 OpenAccess Gear. https://projects.si2.org/openeda.si2.org/projects/oagear

[195] Synopsys Design Compiler User Guide. http://www.synopsys.com

[196] Synopsys HSPICE User Guide. http://www.synopsys.com

[197] Synopsys IC Compiler User Guide. http://www.synopsys.com

[198] Synopsys PrimeTime User Guide. http://www.synopsys.com

[199] Tcl/Tk Built-in Commands Manual. http://www.tcl.tk/man/tcl8.4/TclCmd

[200] Tela Innovations. http://www.tela-inc.com

[201] UCSD PROBE website. http://vlsicad.ucsd.edu/PROBE

[202] UCSD SensOpt Leakage Optimizer (A. B. Kahng and S. Kang, 2010-2011).
http://vlsicad.ucsd.edu/SIZING/optimizer.html

[203] *UCSD TritonSizer homepage.*
https://github.com/abk-openroad/TritonSizer

[204] VLSI CAD Bookshelf, A. E. Caldwell, A. B. Kahng and I. L. Markov.
http://vlsicad.eecs.umich.edu/BK

[205] *Yosys Open Synthesis Suite*, http://www.clifford.at/yosys/