UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Vinícius dos Santos Livramento

TIMING OPTIMIZATION DURING THE PHYSICAL SYNTHESIS OF CELL-BASED VLSI CIRCUITS

Florianópolis

2016

Vinícius dos Santos Livramento

TIMING OPTIMIZATION DURING THE PHYSICAL SYNTHESIS OF CELL-BASED VLSI CIRCUITS

Tese de doutorado submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para obtenção do Grau de Doutor em Engenharia de Automação e Sistemas.
Orientador: Prof. Luiz Cláudio Villar dos Santos, Dr. – INE/UFSC
Coorientador: Prof. José Luís Almada Güntzel, Dr. – INE/UFSC

Florianópolis

2016

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Livramento, Vinícius
Timing Optimization During the Physical Synthesis of Cell-Based VLSI Circuits / Vinícius Livramento ; orientador, Luiz Cláudio Villar dos Santos ; coorientador, José Luís Almada Güntzel. - Florianópolis, SC, 2016.
192 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Inclui referências

1. Engenharia de Automação e Sistemas. 2. Automação de Projeto Eletrônico. 3. Síntese Física. 4. Minimização de violações de timing. 5. Relaxação Lagrangeana. I. Villar dos Santos, Luiz Cláudio. II. Almada Güntzel, José Luís. III. Universidade Federal de Santa Catarina. Programa de Pós Graduação em Engenharia de Automação e Sistemas. IV. Título.

Vinícius dos Santos Livramento

TIMING OPTIMIZATION DURING THE PHYSICAL SYNTHESIS OF CELL-BASED VLSI CIRCUITS

Esta Tese foi julgada aprovada para a obtenção do Título de Doutor em Engenharia de automação e Sistemas, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Florianópolis, 02 de Dezembro 2016.

Prof. Daniel Ferreira Coutinho, Dr. – DAS/UFSC
Coordenador

Prof. Luiz Cláudio Villar dos Santos, Dr. – INE/UFSC
Orientador

Prof. José Luís Almada Güntzel, Dr. – INE/UFSC
Coorientador

Banca Examinadora:

Prof. Luiz Cláudio Villar dos Santos, Dr. – INE/UFSC
Presidente

Prof. Fernando Gehm Moraes, Dr. – PUC/RS

Prof. Sérgio Bampi, Dr. – Instituto de Informática/UFRGS

Prof. Eduardo Camponogara, Dr. – DAS/UFSC

Gustavo Reis Wilke, Dr. – Synopsys, Inc.

Prof. Márcio Bastos Castro, Dr. – INE/UFSC

À minha família.

AGRADECIMENTOS

Aos meus pais, Mário e Cristina, pelo amor, carinho e dedicação que nunca faltaram. Também aos meus irmãos, Natália e Vitor, pelo apoio e compreensão. Ao meu orientador, Luiz Cláudio Villar dos Santos, pela excelente orientação e rigor exigido, os quais foram fundamentais para o sucesso desta tese de doutorado. Ao meu coorientador, José Luís Almada Güntzel, que me acompanha desde o período do mestrado, sempre com muita dedicação e confiança. Aos membros da banca, pelo tempo dedicado para uma revisão rigorosa e pelas sugestões que contribuiram com esta tese. Aos colegas do ECL que de alguma forma participaram deste trabalho. Em particular, aos colegas Chrystian Guth e Renan Netto por todo suporte técnico prestado, essencial para o sucesso deste trabalho. À Capes, pelo custeio parcial da execução deste trabalho e pelo custeio do doutorado sanduíche na cidade de Austin, EUA (Processo número: 99999.0062302015-06).

"Tudo se compõe, e se decompõe." (O Jardim do Silêncio, Paulinho Moska).

RESUMO

A evolução da tecnologia CMOS viabilizou a fabricação de circuitos integrados contendo bilhões de transistores em uma única pastilha de silício, dando origem ao jargão *Very-Large-Scale Integration* (VLSI). A frequência-alvo de operação de um circuito VLSI afeta o seu desempenho e induz restrições de *timing* que devem ser manipuladas pelas ferramentas de síntese. Durante a síntese física de circuitos VLSI, diversas técnicas de otimização são usadas para iterativamente reduzir o número de violações de *timing* até que a frequência-alvo de operação seja atingida. O aumento dramático do atraso das interconexões devido à evolução tecnológica representa um dos maiores desafios para o fluxo de *timing closure* de circuitos VLSI contemporâneos. Nesse cenário, técnicas de síntese de interconexão eficientes têm um papel fundamental. Por este motivo, esta tese aborda dois problemas de otimização de *timing* para uma síntese eficiente das interconexões de um circuito VLSI: *Incremental Timing-Driven Placement* (ITDP) e *Incremental Timing-Driven Layer Assignment* (ITLA). Para resolver o **problema de ITDP**, esta tese propõe uma nova formulação utilizando Relaxação Lagrangeana que tem por objetivo a minimização simultânea das violações de *timing* para restrições do tipo *setup* e *hold*. Este trabalho também propõe uma técnica que utiliza multiplicadores de Lagrange como pesos para as interconexões, os quais são atualizados dinamicamente através dos resultados de uma ferramenta de análise de *timing*. Tal técnica realoca as células do circuito por meio de uma nova busca discreta que adota a distância Euclidiana como vizinhança. Para resolver o **problema de ITLA**, esta tese propõe uma abordagem em fluxo em redes que otimiza simultaneamente segmentos críticos e não-críticos, e explora algumas condições de fluxo para extrair as informações de *timing* para cada segmento individualmente, permitindo assim o uso de uma ferramenta de *timing* externa. A validação experimental, utilizando *benchmarks* derivados de circuitos industriais, demonstra a eficiência das técnicas propostas quando comparadas com trabalhos estado da arte.

Palavras-chave: Síntese Física, Timing Closure, Incremental Timing-Driven Placement, Incremental Timing-Driven Layer Assignment, Relaxação Lagrangeana, Fluxo em Redes.

ABSTRACT

The evolution of CMOS technology made possible integrated circuits with billions of transistors assembled into a single silicon chip, giving rise to the jargon Very-Large-Scale Integration (VLSI). The required clock frequency affects the performance of a VLSI circuit and induces timing constraints that must be properly handled by synthesis tools. During the physical synthesis of VLSI circuits, several optimization techniques are used to iteratively reduce the number of timing violations until the target clock frequency is met. The dramatic increase of interconnect delay under technology scaling represents one of the major challenges for the timing closure of modern VLSI circuits. In this scenario, effective interconnect synthesis techniques play a major role. That is why this thesis targets two timing optimization problems for effective interconnect synthesis: Incremental Timing-Driven Placement (ITDP) and Incremental Timing-Driven Layer Assignment (ITLA). For solving the **ITDP problem**, this thesis proposes a new Lagrangian Relaxation formulation that minimizes timing violations for both setup and hold timing constraints. This work also proposes a net-based technique that uses Lagrange multipliers as net-weights, which are dynamically updated using an accurate timing analyzer. The net-based technique makes use of a novel discrete search to relocate cells by employing the Euclidean distance to define a proper neighborhood. For solving the **ITLA problem**, this thesis proposes a network flow approach that handles simultaneously critical and non-critical segments, and exploits a few flow conservation conditions to extract timing information for each net segment individually, thereby enabling the use of an external timing engine. The experimental validation using benchmark suites derived from industrial circuits demonstrates the effectiveness of the proposed techniques when compared with state-of-the-art works.

Keywords: Physical Synthesis, Timing Closure, Incremental Timing-Driven Placement, Incremental Timing-Driven Layer Assignment, Lagrangian Relaxation, Network Flow.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACRONYMS

ASIC	*Application-Specific Integrated Circuit*
CAD	*Computed-Aided Design*
CMOS	*Complementary Metal-Oxide Semiconductor*
DAG	*Directed Acyclic Graph*
EDA	*Electronic Design Automation*
ESL	*Electronic System Level*
HDL	*Hardware Description Language*
HPWL	*Half-Perimeter Wirelength*
IP	*Intellectual Property*
IRP	*Incremental Register Placement*
ITDP	*Incremental Timing-Driven Placement*
ITLA	*Incremental Timing-Driven Layer Assignment*
LCB	*Local Clock Buffer*
LP	*Linear Programming*
LR	*Lagrangian Relaxation*
NLDM	*Non-Linear Delay Model*
RC	*Resistance Capacitance*
RTL	*Register Transfer Level*
SoC	*System-on-Chip*
STA	*Static Timing Analysis*
TDP	*Timing-Driven Placement*
TNS	*Total Negative Slack*
VLSI	*Very-Large-Scale Integration*
WNS	*Worst Negative Slack*

LIST OF SYMBOLS

$\mathcal{C}$: Set of standard cells
$\mathcal{P}$: Set of fixed elements such as I/O pads and macro blocks
V: Set of Vertices, where $V = \mathcal{C} \cup \mathcal{P}$
$\mathcal{N}$: Set of hyperedges, i.e. nets
$\mathcal{S}$: Set of net segments
$\mathcal{L}$: Set of metal layers

c_j: Standard cell c_j belonging to the set $\mathcal{C}$
s_j: Net segment s_j belonging to the set $\mathcal{S}$
x_j: Horizontal coordinate of cell c_j
y_j: Vertical coordinate of cell c_j
h_j: Height of cell c_j
w_j: Width of cell c_j

$\mathcal{I}_j$: Fanin set of integers to identify vertices connected to the input of v_j
$\mathcal{O}_j$: Fanout set of integers to identify vertices connected to the output of v_j
$\mathcal{TS}$: Set of integers to identify vertices that produce a timing startpoint
$\mathcal{TE}$: Set of integers to identify vertices that consume a timing endpoint

P: Clock period to achieve a frequency $\frac{1}{P}Hz$
a_j^L or a_j: Late arrival time at a given timing point j
a_j^E: Early arrival time at a given timing point j
r_j^L or r_j: Late required time at a given timing point j
r_j^E: Early required time at a given timing point j
slk_j^L or slk_j: Late slack at a given timing point j
slk_j^E: Early slack at a given timing point j

$\delta_{i,j}^L$ or $\delta_{i,j}^c$: Late arc delay of a cell from an input i to the output of cell c_j
$\delta_{i,j}^E$: Early arc delay of a cell from an input i to the output of cell c_j
$\sigma_{i,j}^L$ or $\sigma_{i,j}$: Late arc slew of a cell from an input i to the output of cell c_j
$\sigma_{i,j}^E$: Early arc slew of a cell from an input i to the output of cell c_j
σ_j^L: Worst arc slew of cell c_j
σ_j^E: Best arc slew of cell c_j
σ_j^s: Slew of a net segment s_j
$\tau_{i,j}$: Elmore delay for the wire connecting c_i and c_j
$d_{i,j}^L$: Sum of late arc delay $\delta_{i,j}^L$ and Elmore delay $\tau_{i,j}$

$d^E_{i,j}$: Sum of early arc delay $\delta^E_{i,j}$ and Elmore delay $\tau_{i,j}$

$R^s_j(q)$: Resistance of segment s_j at layer l_q
$R^v(q)$: Resistance of via between two consecutive layers l_q and l_{q+1}
$C^s_j(q)$: Capacitance of segment s_j at layer l_q
$C^v(q)$: Capacitance of via between two consecutive layers l_q and l_{q+1}
C^{down}_j: Downstream capacitance of a cell, a net segment, or a via
C^{in}_j: Input capacitance of c_j

$\delta^s_j(q)$: Delay of a net segment s_j assigned to a layer l_q
$\delta^v(q)$: Delay of via between two consecutive layers l_q and l_{q+1}
$\delta^v_{i,j}(p,q)$: Delay of via connecting the net segments s_i and s_j at layers l_p and l_q

$\lambda^L_{i,j}$: Late net-weight from the output of c_i to the output of c_j
$\lambda^E_{i,j}$: Early net-weight from the output of c_i to the output of c_j
$\lambda^c_{i,j}$: Lagrange multiplier associated with a timing arc i,j of a cell c_j
λ^s_j: Lagrange multiplier associated with a net segment s_j

$\alpha_{j,q}$: Binary decision variable equals 1 if segment s_j is assigned to layer l_q

CONTENTS

1 INTRODUCTION

This chapter first summarizes the overall hardware design flow when targeting cell-based VLSI designs (e.g. Systems-on-Chip, Application-Specific Integrated Circuits). Then it focuses on the physical synthesis phase and points out the challenging timing optimization problems tackled in the thesis. The chapter ends with a discussion of the scientific contributions.

1.1 REQUIREMENTS AND CHALLENGES OF MODERN VLSI CIRCUITS

The astonishing evolution of CMOS technology made possible integrated circuits with hundreds of millions of transistors (or even billions) assembled into a single silicon chip, giving rise to the jargon Very-Large-Scale Integration (VLSI) (KAHNG et al., 2011). VLSI systems span a wide range of classes of integrated circuits such as high-end microprocessors, Field-Programmable Gate Arrays (FPGAs), Application-Specific Integrated Circuits (ASICs), and Systems-on-Chip (SoCs) (PAPA, 2010). The design of a VLSI circuit is a highly complex process that can be carried out by two main circuit-design styles adopted in the industry: full-custom and semi-custom. Full-custom is a labor-intensive methodology where the component layouts are handcrafted and placed almost anywhere in the layout surface to obtain a very compact chip with optimized electrical properties (SAIT; YOUSSEF, 1999; KAHNG et al., 2011). This design style targets the specific class of circuits where the design cost can be amortized over high-volume production such as high-end microprocessors and FPGAs. Semi-custom (or Cell-Based) is a highly-automated design methodology that relies on predesigned elements to reduce the design complexity and hence, the overall cost and time-to-market. This design style targets a different market segment that includes ASICs and SoCs. Although cell-based circuits are slower and consume more power than full-custom circuits, this gap has been decreasing over the last years thanks to the constant research and innovation in the area (CHINNERY, 2013).

SoCs are very complex heterogeneous systems that are composed of one or more CPUs, hardware accelerators, memory subsystem (with one or more levels of cache), interconnect fabric, and non-volatile memory to store the embedded software (MARTIN; CHANG, 2003). Figure

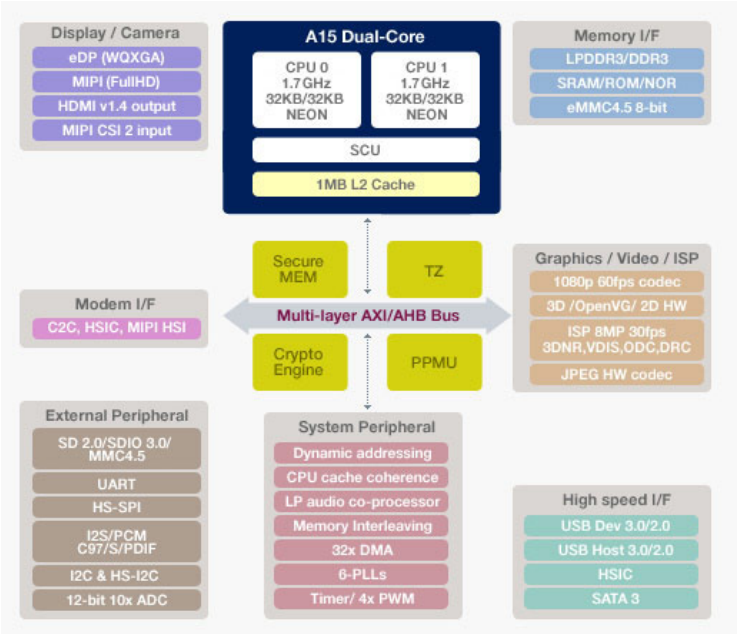


Figure 1: **Example of an SoC: Samsung Exynos 5 Dual.** *A couple of smartphones that use this SoC are Google Nexus 10 and Samsung Galaxy Tab II. Obtained from (SAMSUNG, 2016).*

1 shows the block diagram of a contemporary SoC that uses a pair of ARM Cortex-A15 CPUs.

Nowadays, SoCs represent the main drivers for cell-based design methodologies as they are responsible for fueling the consumer electronics market, especially in the segment of smartphones, tablets, and other portable devices like digital cameras and gaming consoles (ITRS, 2015). Figure 2 brings the IC market share and growth forecast from 2013 to 2018 for different classes of products. Observe that products using SoCs account for a large portion of the market (e.g. cellphones, tablets, game consoles). Also notice that, although the growth forecast for smartphones and tablets is smaller than other segments (e.g. Internet of Things) they will keep growing and will represent an important share of the IC market. The very short product life cycle of consumer electronics and the growing demand for functionality and high-performance create the need for continued research for new algorithms and methodologies.

The consumer electronics market has been pushing SoC power and performance requirements and represents a challenge for the indus-

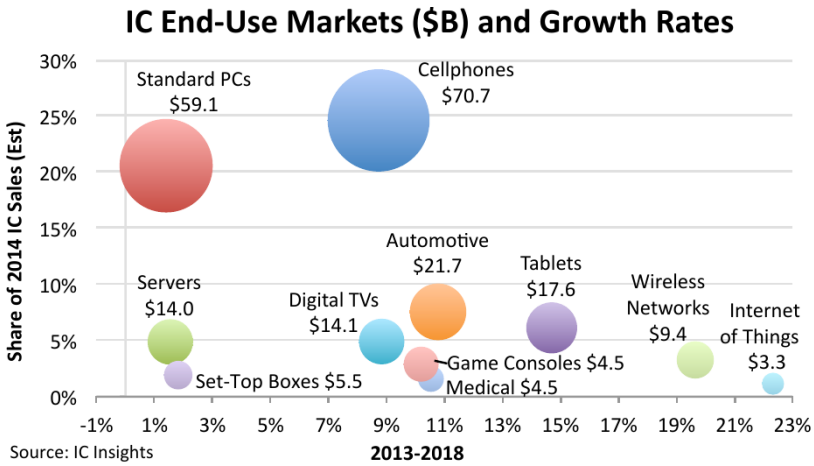


Figure 2: **Market share of 2014 IC sales (y-axis) and projected growth in IC sales from 2013 to 2018 (x-axis).** *Cellphones continue to be the largest drivers of IC sales, accounting for 25% of the total, while standard PCs represent 21%. Other produts following cell-based methodology like tablets and game consoles also account for an important market share and will continue growing. Obtained from (INSIGHTS, 2014).*

try: the execution of computationally-intensive applications using the least amount of energy (to maximize battery life) and the compliance with a power limit (to enable power dissipation from a compact device) (CHAN et al., 2014).

The **required clock frequency** affects the performance of a VLSI circuit and induces timing constraints that must be properly handled by synthesis tools. This thesis focuses on techniques for **timing closure of cell-based VLSI circuits**, i.e. techniques able to iteratively reduce the number of timing violations until the synthesis of the synchronous digital system reaches the specified target frequency.

1.2 DESIGN FLOW OF CELL-BASED VLSI CIRCUITS

The design of VLSI circuits is a very complex task that demands the use of sophisticated tools and methodologies that are referred to as Electronic Design Automation (EDA) (PAPA, 2010; KAHNG et al., 2011). To cope with the design complexity and to reduce the design time, the vast majority of hardware design flows use libraries of

predesigned components (the so-called standard cells) and intellectual property (IP) blocks. A standard cell library contains the physical layout of primitive building blocks with different functionalities (e.g., logic gates, adders, registers), which are provided by foundries or vendors (CHINNERY; KEUTZER, 2002). For each standard cell, the library provides the geometric information (e.g., layout, width, height) and the electrical characteristics (delay, power, etc) that are required for successful hardware synthesis.

The first step of an SoC design flow defines the system specification and performs the hardware/software partitioning at Electronic System Level (ESL). From the system specification, ESL tools help define the **hardware specification** and the requirements such as performance, power, and area, as well as the basic architecture, including memories and IP blocks, which represent the starting point for the hardware design flow (MARTIN; BAILEY; PIZIALI, 2010).

Figure 3 illustrates a simplified top-down hardware design flow[1]. It starts at a high abstraction level, defining the hardware specifications and general architecture, and it ends at a low abstraction level, where the chip layout geometry is determined for fabrication (COUSSY; MORAWIEC, 2010). EDA tools are adopted in almost every step of the design flow. The **functional design** step describes the functionality and timing behavior at register transfer level (RTL) using hardware description languages (HDL) such as *VHDL* and *Verilog*. The **logic synthesis and technology mapping** step is an automated process that converts the described hardware into boolean expressions and maps them into a set of standard cells from a library. Then **physical synthesis** (also known as physical design) instantiates the layout of standard cells (their geometric representations), assigns their spatial locations, and creates the connections between them. **Physical verification** checks the electrical and logic functionality of the physical layout while **signoff** validates the circuit timing and fixes some minor errors. In the **fabrication** step, the layout is sent to a foundry to be manufactured. Finally, a **packaging and testing** step prepares the fabricated circuit for actual use.

In earlier technology generations (say, above 180 nm), the delays of logic gates were a determinant factor of circuit timing, whereas resistance and capacitance of interconnections (wires) were proportionally less significant and could be disregarded. In such scenario, logic synthesis was able to provide accurate timing estimates (ALPERT et al.,

[1] Although the hardware design flow is iterative, this is not captured in the figure for simplicity. Besides, a few steps are merged while others are omitted for clarity.

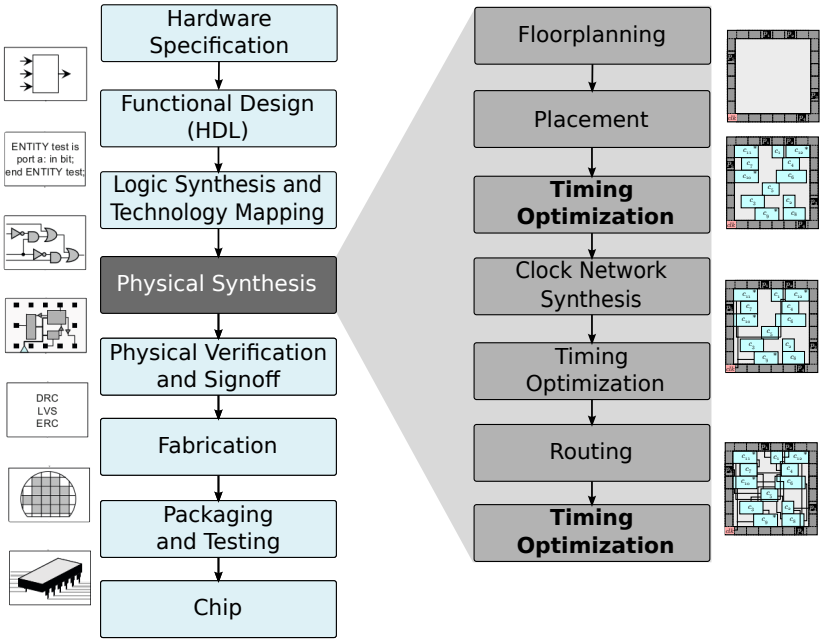


Figure 3: **Major steps of hardware design flow.** *The right-hand side illustrates the main physical synthesis steps. Adapted from (KAHNG et al., 2011).*

2007; ALPERT; CHU; VILLARRUBIA, 2007). However, in technologies below 250 nm, interconnections became more resistive and thus their delays are much more significant as compared to the delays of logic gates (HO; MAI; HOROWITZ, 2001; SAXENA et al., 2004). As a consequence, timing estimates without physical information would be so inaccurate to the point that the physical design steps could invalidate the optimizations obtained during logic synthesis (PAPA et al., 2011). Figure 4 shows the dramatic increase of global interconnect delay with technology scaling. Being up to 3 orders of magnitude higher than gate delay for 32nm, it represents one of the major challenges to be overcome during the physical synthesis of modern VLSI circuits. In face of this trend, **physical synthesis** became again an intensive research field (ALPERT; CHU; VILLARRUBIA, 2007).

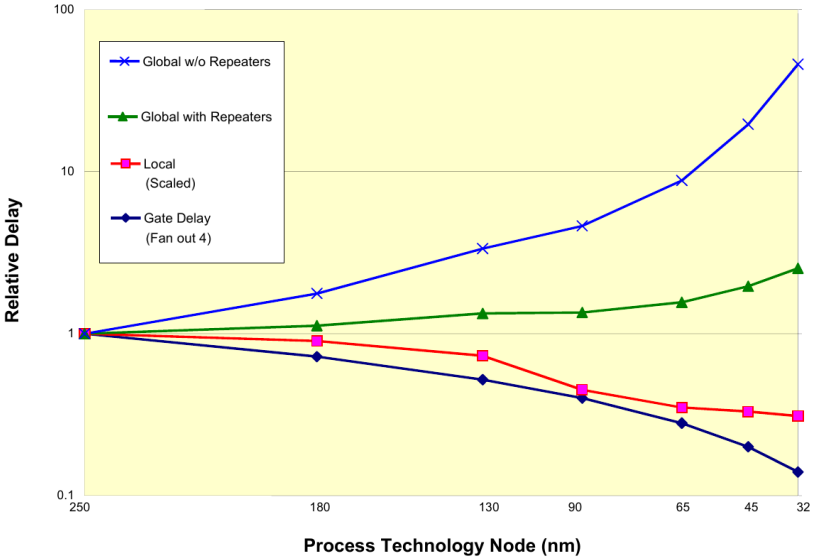


Figure 4: **Gate and interconnect delay versus technology scaling.** *Local wires correspond to Metal 1 level (intra-cell connections), while global wires correspond to upper metal levels (inter-cell connections). Notice that although local interconnect and gate delays have scaled down, global interconnect delays have grown (even with repeaters) and represent a bottleneck for circuit timing. Obtained from (ITRS, 2005).*

1.3 PHYSICAL SYNTHESIS

The main objective of physical synthesis is to obtain a circuit layout that satisfies the timing constraints induced by the target clock frequency (VISWANATHAN et al., 2010; KAHNG, 2015). During the physical synthesis, several optimization techniques are used to gradually reduce the number of timing violations[2] until the circuit reaches the specified clock frequency. Such an iterative process, known as timing closure, corresponds to the most critical task of modern SoCs (ALPERT et al., 2007; KAHNG, 2015)

This section first reviews the main steps of physical synthesis. Then it reviews the timing closure flow and points out some important challenges faced by contemporary physical synthesis. The section ends

[2]Timing violations represent a measure of how far from the timing constraints (specified to achieve the target frequency) the circuit is.

up with a description of the problems tackled in the thesis.

1.3.1 Main Steps

As illustrated on the right-hand side of Figure 3, physical synthesis consists of the following steps:

- **Floorplanning** defines the chip dimensions and locations of I/O pins and macro blocks (e.g., memories and intellectual property blocks) and defines how to partition the circuit into smaller subcircuits, when possible;
- **Placement** finds the planar locations of standard cells so as to minimize an estimate of the wirelengths between the cells and keep the density[3] profile under control;
- **Clock Network Synthesis** determines the clock network topology to deliver the clock signal to each sequential[4] element and also adds buffers in some specific parts of the clock network so as to meet skew[5] and delay requirements;
- **Routing** connects the instantiated standard cells using different metal layers so as to minimize the wirelength and comply with limited routing resources;
- **Timing Optimization** techniques are used between different steps of physical synthesis to gradually achieve the circuit timing constraints. The earlier in the design flow, the more opportunities for changes, but the lower the accuracy; the later in the design flow, the higher the accuracy, but the more limited the amount of changes.

During the Placement and Routing steps, the primary goals are to minimize the interconnect wirelength and comply with some design

[3] Density gives an indication of the average standard cell utilization over the whole chip area. Generally, the chip area is divided into squares and there is a density constraint, say 0.7, which indicates that the average standard cell area over the avaible area on each square is below 0.7. It is important to keep density under control to ensure that the routing step can provide an effective solution (KIM et al., 2012). More details are given in Section 2.1.4.

[4] A storage element controlled by the clock signal such as a register or a latch.

[5] The skew is the maximum delay difference of the clock signal between any pair of sequential elements.

constraints, while timing is generally considered as a secondary objective metric (ALPERT; MEHTA; SAPATNEKAR, 2008). That is why incremental optimization techniques play a major role during physical synthesis to improve previously optimized solutions (COUDERT et al., 2000). This thesis targets **two** incremental timing optimization problems at different steps of the physical synthesis flow (as highlighted on the right-hand side of Figure 3). The **first** is solved right after the Placement step, when the planar locations of the cells were already defined. Although this step of the design flow provides more opportunity for changes, it relies on wirelength estimates since the actual routing has not yet been defined. The **second** problem is tackled right after the Routing step, when the interconnection topologies and metal layers were already defined. This step provides much higher accuracy for the interconnect delays, but the opportunity for changes is smaller. The next two subsections provide illustrative examples to show the importance of each of the two timing optimization problems.

1.3.2 The Impact of Placement on Timing

Placement is a key step to physical synthesis and has a huge impact on circuit timing due to four main reasons (WANG; CHANG; CHENG, 2009) (PAPA et al., 2011):

1. The cell locations have a strong impact on the interconnections wirelengths, which affect signal delays due to their resistance and capacitance.

2. The locations of sequential elements determined by the placement step serve as the starting point for clock network synthesis (therefore, a poor placement can undermine the quality of the clock network).

3. The distribution of cells along the planar surface has a substantial impact on the routing step (therefore a well-placed circuit will have shorter interconnections).

4. The density of cells has a direct relation with heat distribution throughout the chip and therefore affects power consumption.

Figure 5 illustrates the impact of placement through a toy example. Note that the interconnections are longer for the solution on

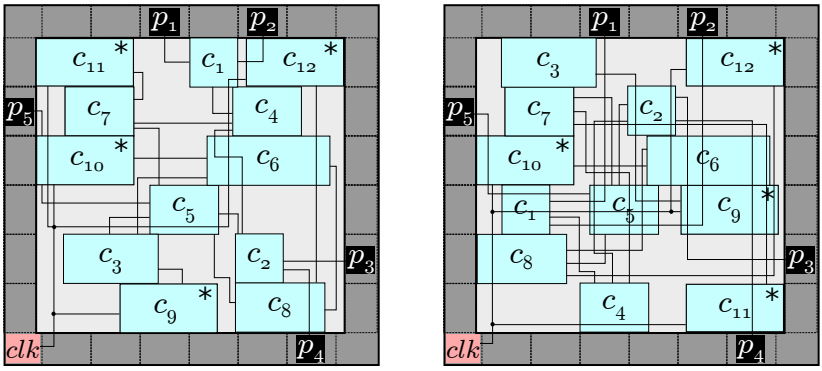


Figure 5: **Impact of placement.** *The symbols c and p represent standard cells and circuit pads (i.e. an electrical terminal used as a connection for external inputs and outputs), respectively. To mark the cells consisting of a sequential element, we employ an asterisk. Comparing the alternative placement solutions, observe that the poor placement on the right-hand side dramatically increases the circuit wirelength and, therefore degrades circuit timing and routability.*

the right-hand side. This illustrates how cell locations influence the resulting wirelength and, therefore, circuit timing. For this reason, techniques that optimize cell locations are likely to drive the next generation of research on physical synthesis (ALPERT et al., 2012; MARKOV; HU; KIM, 2012).

1.3.3 The Impact of Routing on Timing

The global routing step has a major impact on circuit timing as it defines the interconnection topologies and the metal layers (HELD et al., 2015). Modern technologies provide several metal layers with different widths and thicknesses, where upper layers are wider and thicker than lower ones, as illustrated in Figure 6(a). Figure 6(b) illustrates the impact of routing through a toy example. Note that the interconnections are much longer for the solution on the right-hand side. In addition, the critical net (n_2) is routed in the lowest layer, which is much more resistive than the upper layer, degrading the net delay. This illustrates how the choice of interconnect topology and routing layer affects the circuit timing.

Technology nodes below 32 nm provide between four to six different metal widths and thicknesses where upper layers can be up to 20

times thicker than lower ones (ALPERT et al., 2010). Despite the dramatic reduction of resistance on upper layers, they require more area and thus offer less resources for routing. Therefore, upper metal layers must be wisely used to reduce the delay of critical interconnects. For this reason, efficient interconnect synthesis techniques are of utmost importance for the timing closure of contemporary circuits (LI et al., 2008; HU; LI; ALPERT, 2009; WEI et al., 2013).

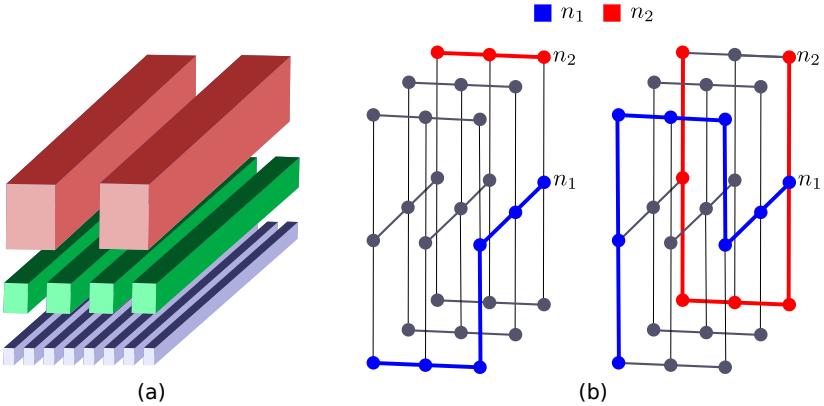


Figure 6: **Impact of routing.** *(a) 3 groups of metal layers with different widths and thicknesses. (b) 3x3 routing grids with 3 metal layers and 2 nets. Comparing the alternative routing solutions, observe that the poor routing on the right-hand side introduces a lot of detours, dramatically increasing the interconnect wirelengths and also degrading the circuit timing.*

1.3.4 Timing Closure

Timing closure of modern circuits is not a straightforward task that can be easily achieved by a simple top-down process. In practice, it is hardly achieved in a single iteration (PAPA et al., 2011). An industrial timing closure flow is an iterative process that is integrated to several steps of physical synthesis, as illustrated in Figure 7.

Observe that, at different stages of the design flow, there are decision steps to check for timing closure. During the timing closure steps, a timing analysis engine is employed (for the timing violation analysis) to assess whether the target clock frequency is met or not. In other words, it determines the difference between the clock period and how long a signal takes to traverse the longest path (PAPA, 2010).

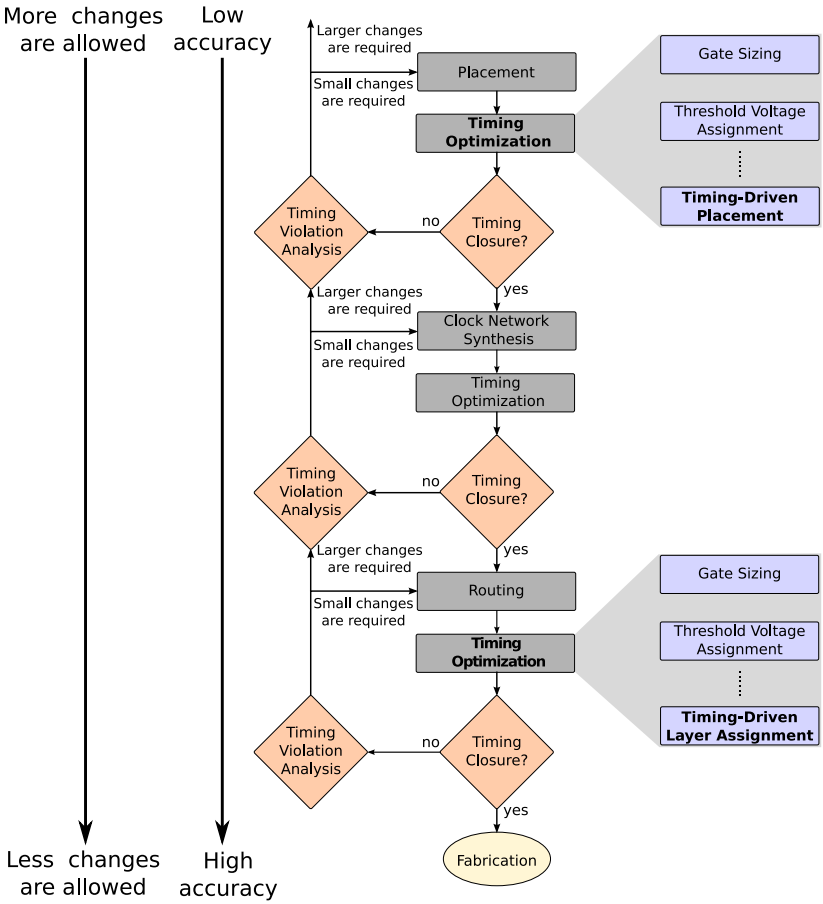


Figure 7: **Timing closure during physical synthesis.** *Although the design flow varies from vendor to vendor, this flowchart captures the iterative nature of timing closure process (GREG, 2016).*

Failing to achieve timing closure in a given step means that designers must analyze the violations reported by the timing analysis engine, identify the failing steps, and backtrack in the design flow. Therefore, the physical synthesis turnaround time strongly depends on the success of timing closure.

Notice that the design flow enforces gradual reduction in the amount of changes to enforce convergence. The timing models also become more accurate as the flow progresses. For example, after the placement step, although optimization techniques rely on wirelength estimates (since routing information is not yet available) many changes can be made to improve timing. Therefore, at the beginning of the design flow, the timing closure step also checks the timing constraints assuming approximative timing models. As the design converges to final steps, although the accuracy of timing information increases, the scope and magnitude of changes are reduced. Therefore, it is expected that the amount of violations in late steps is small enough. Failing to achieve timing closure in late steps may require to backtrack to the placement, or even to the logic synthesis step, which can undermine the tight design schedule, resulting in a late time-to-market, which may lead to millions of dollars of loss in revenue.

During physical synthesis, timing optimization techniques such as gate sizing and threshold voltage assignment are widely used to modify the electrical characteristics of standard cells so as to improve timing. These techniques take advantage of the fact that a standard cell library provides, for each cell, many implementation options with different timing and power characteristics (OZDAL; BURNS; HU, 2012). Being limited to exploiting cell characteristics, these techniques are not able to modify the interconnection characteristics to optimize the circuit timing. In the scenario of contemporary designs, where the interconnections play a very important role, **timing-driven placement** and **timing-driven layer assignment** appear as promising optimization techniques, and therefore, attracts the attention from both industry and academia (ALPERT et al., 2012; KIM; HU; VISWANATHAN, 2014; YU et al., 2015).

Although there has been research in timing optimization techniques for more than 20 years, most of them were conceived for old technological scenarios. Since the evolution and changes in the semiconductor industry is very rapid, new research and techniques must consider the challenges to be tackled for contemporary physical synthesis. The next subsection presents a set of challenges to be tackled by optimization techniques in contemporary physical synthesis, accord-

ing to the EDA industry.

1.3.5 Challenges for Contemporary Physical Synthesis

The continuous evolution of technology brings new challenges that require the EDA industry to review many supposedly consolidated solutions. Some of the challenges are detailed below.

- **Heterogeneous fabrics:** The high-complexity and short time-to-market requires the use of hierarchical methodologies with thirdy-party macro (IP) blocks. This demands an heterogeneous integration of standard cells and macro blocks, causing serious challenges to circuit legalization[6] and timing closure (PAPA et al., 2011; ALPERT; MEHTA; SAPATNEKAR, 2008).

- **Fast turnaround time:** The time-to-market pressure leads to very fast turnaround time requirements. Recent papers from the industry point out a very tight turnaround time of 12 hours per million of cells for the whole physical synthesis (PAPA et al., 2011; REIMANN; SZE; REIS, 2016). Therefore, only near-linear-time optimization algorithms can be applied to fit in that runtime budget.

- **Problem size:** The number of cells in modern circuit grows at a steep rate so as to address the increasing demand for new functionalities. Although hierarchical approaches can alleviate the problem complexity, modern optimization techniques must be scalable enough to handle subcircuits with millions of cells (ALPERT; MEHTA; SAPATNEKAR, 2008).

- **NP-complete problems:** A number of optimization problems to be solved during physical design belong to the NP-complete[7] class (PAPA et al., 2011). In short, very large instances of NP-complete problems must be solved quickly to cope with the runtime budget required by contemporary physical synthesis. This renders the need of very efficient algorithms and heuristics that

[6] A step performed to ensure that cells and macro blocks do not overlap and satisfy some design rules required for the circuit fabrication.

[7] It refers to the class of problems for which the solution cannot be found in polynomial time w.r.t. the number of inputs, but it can be verified in polynomial time (CORMEN et al., 2009). For example, the solution of timing optimization problems, which are the scope of this thesis, can be verified in polynomial time using a static timing analysis tool.

must take advantage of problem specific characteristics and technology parameters.

Therefore, the proposed solutions for the problems tackled in this thesis take into account the aforementioned challenges[8].

1.3.6 Problems Tackled in the Thesis

Recent position papers from industry and academia point out the importance of efficient interconnect synthesis techniques to satisfy the circuit timing constraints and close on timing (ALPERT et al., 2012; MARKOV; HU; KIM, 2012; LI et al., 2012; WEI et al., 2013). Therefore, this thesis focuses on specific optimization techniques that have a direct impact on circuit interconnects, one applied right after the global placement step and another applied right after the global routing step (as highlighted in Figure 7). The specific motivations and definitions for the **two** problems tackled in the thesis are detailed in the following:

1. **Incremental Timing-Driven Placement (ITDP)** finds new locations for standard cells (both logic and sequential cells) so as to improve the circuit timing and preserve the placement quality in terms of wirelength and density. Figure 8 illustrates the basic idea of ITDP through a toy example. Although there have been significant advances in ITDP, there is a lack of efficient techniques to handle the growing number of cells and macro blocks in current and future designs. This problem is addressed in Chapter 3.

2. **Incremental Timing-Driven Layer Assignment (ITLA)** reassigns the metal layers of critical and non-critical net segments so as to improve the circuit timing while observing the routing capacity constraints. Figure 9 illustrates the basic idea of ITLA through a toy example. Although many timing-driven layer assignment techniques can be found in the literature, most were conceived with inaccurate timing models, which is not appropriate for late stages of the physical design flow. This problem is addressed in Chapter 4.

[8] Other important challenges like complex manufacturing rules for advanced technology nodes can also be considered (XU et al., 2016).

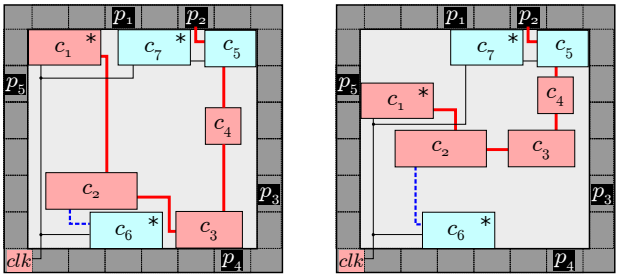


Figure 8: **Basic idea of incremental timing-driven placement**. *Thick (red) wires on the left-hand side solution indicate those that most affect the circuit timing and thus should be shortened as much as possible. The idea of incremental timing-driven placement is to relocate some cells to shorten those critical wires without increasing too much the length of the others. The alternative solution on the right-hand side relocates cells c_1, c_2, c_3, and c_4 to shorten those critical wires so as to reduce the signal delay and improve the circuit timing. Note that care must be taken to preserve the placement quality. For instance, avoiding the introduction of significant overhead in the wires connected to other cells, such as the wire from c_2 to c_6 (dotted blue line).*

1.4 SCIENTIFIC CONTRIBUTIONS

The solutions presented in this thesis to solve the two problems listed in the previous subsection bring the following innovations that represent the main scientific contributions:

1. **A novel incremental timing-driven placement formulation based on Lagrangian Relaxation**: A new Lagrangian Relaxation formulation for ITDP that minimizes the total negative slack for both setup and hold timing violations, where Lagrange multipliers are used as net weights and are dynamically updated with an accurate timing analyzer. To solve the formulation, this work proposes a technique that relies on a novel discrete search and employs Euclidean distance to define a proper neighborhood. To further improve circuit timing, we also propose a technique to exploit non-critical interconnect branches and to reduce the capacitive load of critical cells. Discussion of detailed contributions in face of related works will be addressed in Chapter 3.

2. **A novel incremental layer assignment technique driven**

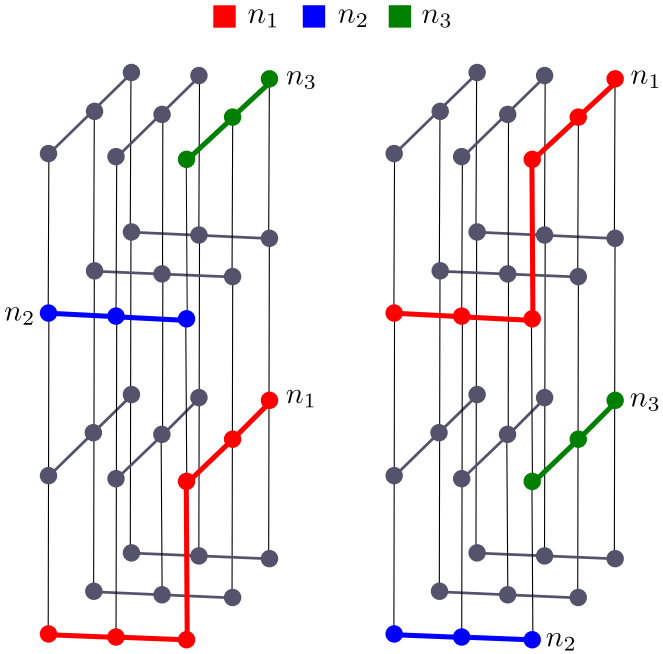


Figure 9: **Basic idea of incremental timing-driven layer assignment.** *A 3x3 routing grid with 4 layers and 3 nets: one critical (n_1) and two non-critical (n_2 and n_3). The idea of incremental timing-driven layer assignment is to re-assign the layers of critical and non-critical nets so as to improve the circuit timing while satisfying routing constraints. The alternative solution on the right-hand side re-assigns non-critical nets to release upper layers for the critical net. Care must be taken to satisfy the routing capacity constraints.*

by an external signoff timing engine[9]: The new approach handles simultaneously critical and non-critical segments and Karush-Kuhn-Tucker to extract timing information for each net segment individually, thereby enabling the use of an external timing engine. Discussion of detailed contributions in face of related works will be addressed in Chapter 4.

[9] A timing analysis tool that employs high-order models to obtain accurate delay estimates (KAHNG et al., 2011).

1.5 PUBLICATIONS, AWARD, AND DISTINCTION

The results of the research described in this thesis were previously reported in four conference papers and two articles, as follows:

- **Timing-Driven Placement Based on Dynamic Net-Weighting for Efficient Slack Histogram Compression** published in the proceedings of the ACM International Symposium on Physical Design[10] (GUTH; LIVRAMENTO et al., 2015).
- **Exploiting Non-Critical Steiner Tree Branches for Post-Placement Timing Optimization** published in the proceeding of the ACM/IEEE International Conference on Computer-Aided Design (LIVRAMENTO et al., 2015).
- **Speeding up Incremental Legalization with Fast Queries to Multidimensional Trees** published in the proceedings of the IEEE Computer Society Annual Symposium on VLSI (NETTO; LIVRAMENTO et al., 2016b).
- **Evaluating the Impact of Circuit Legalization on Incremental Optimization Techniques** published in the proceedings of the IEEE Symposium on Integrated Circuits and Systems Design (NETTO; LIVRAMENTO et al., 2016a)
- **Clock-Tree-Aware Incremental Timing-Driven Placement** published in the ACM Transaction on Design Automation of Electronic Systems (LIVRAMENTO et al., 2016).
- **Incremental Layer Assignment Driven by an External Signoff Timing Engine** published in the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (LIVRAMENTO et al., 2017).

Besides, the implementations of the proposed techniques were submitted to international contests on Computer-Aided Design (CAD) to allow a direct comparison of their results with other groups worldwide. This effort resulted in one award and one distinction.

- First place in the ACM/SIGDA ICCAD Contest on Incremental Timing-Driven Placement 2015 among 42 teams (KIM et al., 2015).

[10] This paper was among the three candidates for best paper award.

- Fifth place in the ACM/SIGDA ICCAD Contest on Incremental Timing-Driven Placement 2014 among 27 teams (KIM; HU; VISWANATHAN, 2014).

1.6 REPRODUCIBILITY

To allow for reproducibility of the proposed techniques reported in this thesis, the algorithms are fully described in pseudo-code and the experimental infrastructure are available in the public domain. The experimental parameters adopted on each algorithm are also reported.

1.7 ORGANIZATION OF THIS THESIS

The rest of this thesis is organized as follows. Chapter 2 presents enough fundamental concepts to keep this thesis self-contained. Chapters 3 and 4 present the solution techniques to address the target problems. Each of the chapters identifies the contribution of proposed techniques in face of related works, reports experimental results, and discusses partial conclusions. Finally, Chapter 5 draws the overall conclusions and perspectives.

2 FUNDAMENTAL CONCEPTS

To keep this thesis self-contained, this chapter presents the fundamental concepts needed to understand the next chapters. First, it introduces basic placement concepts, terminology, and modeling. Then it formalizes timing modeling. Finally, it reviews the basic concepts of Lagrangian Relaxation.

2.1 PLACEMENT GOALS AND CONSTRAINTS

This section first reviews the main concepts associated with designs that use standard and macro cells as the building blocks of circuits' layouts. It briefly describes how placement tools produce an **abstraction** of the actual layout where circuit components are assigned to locations, but whose actual interconnections will be defined later on during the upcoming routing step. It discusses how **estimates** of the actual interconnection wirelength can be obtained for guiding placement. This is because the circuit wirelength gives a reasonable first-order approximation of real objective functions like timing, power, and routability (ALPERT; MEHTA; SAPATNEKAR, 2008).

2.1.1 Cell-Based Design

A standard cell corresponds to a predesigned block composed of a shape, a set of electrical characteristics, and a layout. During physical synthesis, EDA tools handle the external attributes of cells (front-end view): the shape (height and width) of the envelop containing the layout and the set of electrical characteristics (delay, power etc). The inner layout (back-end view) relates to the geometries of the various materials and/or steps that are used by foundries for fabrication. Standard cells follow a restricted layout style so as to reduce the design complexity and therefore the time-to-market (KAHNG et al., 2011). All cells have a fixed height and must be placed in rows to be connected to power and ground supply rails, as exemplified by the inverter layout shown in Figure 10.

The floorplanning step defines the chip physical boundaries (left, right, bottom, and top), denoted as $X_{left}, X_{right}, Y_{bottom}$, and Y_{top}. It also defines the placement area, which is divided into a grid of standard

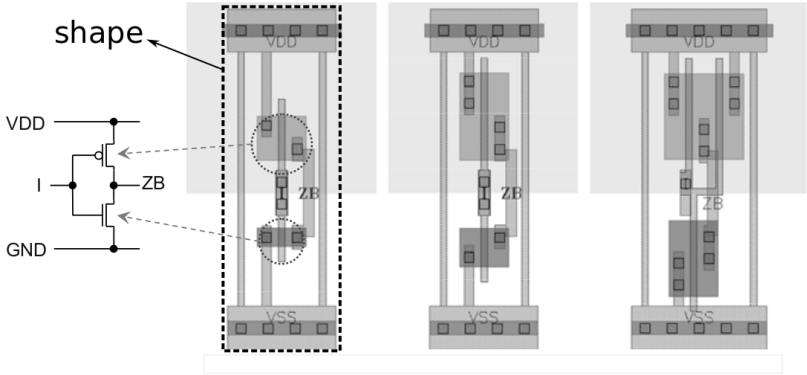


Figure 10: **Layout examples with different electrical characteristics and shapes.** *All layouts have the same functionality (an inverter) and are built for the same technology (65 nm). The leftmost layout corresponds to the slowest inverter; the rightmost, to the fastest. Observe that the three options have exactly the same height, although the width can vary from one option to another. Adapted from (KAHNG et al., 2011).*

cells rows (horizontal) and sites (vertical) that are used for wiring during the routing step, as exemplified in Figure 11. Therefore, the height of cells must be a multiple of the row height; the width, a multiple of site width. The routing between cells can only be performed in horizontal and vertical directions, also known as Manhattan routing. The locations of the macro blocks are also defined during the floorplanning step[1]. Macro blocks are large pieces of reusable logic (also known as IP blocks) that are employed to reduce the design cost and complexity. A macro block can be a processor core, a memory block, or even a video hardware accelerator (as in the SoC from Figure 1). These blocks can have very different shapes and sizes and are treated as blockages during the placement of standard cells.

Placement is generally performed in two phases. The **first phase**, which is called global placement, aims to put the standard cells within the placement area assuming cells as dimensionless elements (i.e. points). Global placement is generally performed using analytical techniques, which try to minimize the total circuit wirelength by solving a huge system of linear equations (KAHNG et al., 2011). During global place-

[1] Although there are a few works that move macro blocks during the placement stage (known as mixed-size placement) (KIM et al., 2012), as a general rule the macros are placed during the floorplanning step and remain fixed for the rest of the design flow.

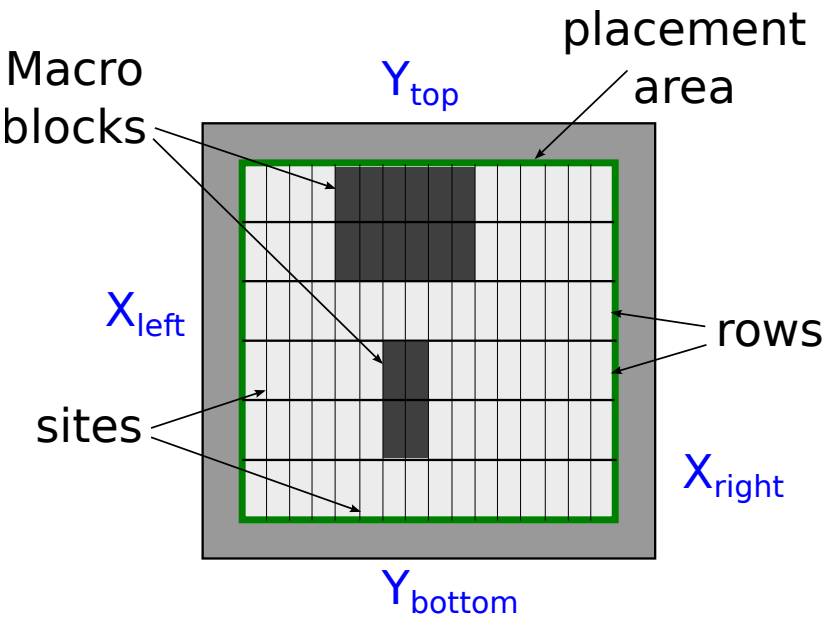


Figure 11: **Circuit boundaries and grids.** *The thick rectangle (green) identifies the boundaries of the cells' placement area. The gray ring is reserved for I/O pads.*

ment, the alignment to rows and sites is ignored, as well as overlaps between cells. Then there is a **second phase** called legalization (also known as detailed placement), which aligns all cells to rows and sites, removes overlaps, and tries to maintain the wirelength from the first phase. Figure 12 depicts examples of movements to align cells to sites and rows, as well as movements to remove horizontal and vertical overlaps.

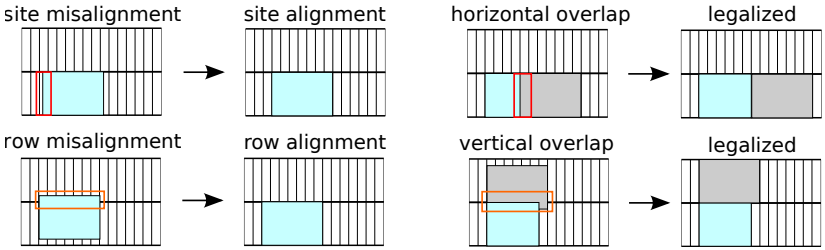


Figure 12: **Examples of standard cell alignment and legalization.**

2.1.2 Abstract Design Representation

Placement techniques generally represent the circuit as a hypergraph $G(V,\mathcal{N})$. The vertex set is composed of two disjoint subsets $\mathcal{C}$ and $\mathcal{P}$, i.e., $V = \{\mathcal{C} \cup \mathcal{P}\}$ with $\mathcal{C} \cap \mathcal{P} = \emptyset$. The elements of subset $\mathcal{C} = \{c_1, c_2, ..., c_n\}$ represent standard cells while the subset $\mathcal{P} = \{p_1, p_2, ..., p_n\}$ contains pre-placed elements such as macro blocks and I/O pads, which have fixed locations and cannot be moved. The set of hyperedges $\mathcal{N} = \{n_1, n_2, ..., n_m\}$ represents the nets (wires) connecting two or more vertices (ALPERT; MEHTA; SAPATNEKAR, 2008). For each $c_i \in \mathcal{C}$, its attributes w_i and h_i denote the width and the height dimensions of the corresponding cell, while the pair (x_i, y_i) denotes its location, expressed by the coordinates of the cell's bottom left corner.

Let us sketch a formulation for placement as an optimization problem: given the hypergraph $G(V,\mathcal{N})$, find a location (x_i, y_i) for every $c_i \in \mathcal{C}$ so as to optimize some design goal(s). Depending on the design requirements, placement may have distinct objectives such as minimizing wirelength, timing violations, or improving routability. Although placement is in general a multi-objective optimization problem, even a single-objective instance of it is NP-hard[2] problem. Therefore, the direct handling of multiple objectives is not viable. That is why, in practice, placement tools usually try to minimize the wirelength, a simple metric that correlates quite well with timing, power, routability, among others (ALPERT; MEHTA; SAPATNEKAR, 2008).

2.1.3 Wirelength Estimation

Since routing information is not yet available during the placement step, optimization techniques rely on interconnection estimates as a metric to minimize the total circuit wirelength (SPINDLER, 2008). There are several models that can be used during placement to estimate interconnections, each with different accuracy, footprint and runtime cost. Figure 13 presents four different models. The most accurate model is the Steiner tree due to its good correlation with the final routing (OBERMEIER; JOHANNES, 2004). However, using the Steiner tree model can be computationally expensive and the Half-Perimeter

[2]It refers to the class of problems for which the optimal solution cannot be found nor verified in polynomial time with respect to the number of inputs. In other words, the time complexity cannot be defined as $O(n^k)$, where n is the number of inputs of the problem and k is a constant (CORMEN et al., 2009).

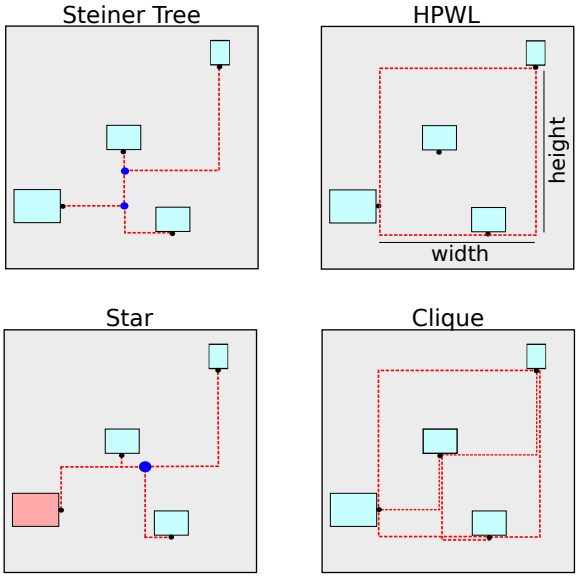


Figure 13: **Different net models that can be used during placement.** *The dots (in blue) along the net in the Steiner model represent the Steiner points. The blue dot in the Star model represents a star point (i.e. the interconnection center of mass).*

Wirelength (HPWL) model is generally used instead (due to its reasonable accuracy and low computational cost). HPWL estimates the wirelength through the half-perimeter of the rectangle that encloses all cell pins in a given interconnection. It is as accurate as the Steiner tree model for nets involving two or three pins, but underestimates the length for nets with four or more pins (KAHNG et al., 2011). The star model connects all interconnect pins with the interconnect center of mass while the clique model connects all pairs of pins. Both star and clique models overestimate the interconnect wirelength but are often preferred in analytical placement tools[3], although they are less accurate than HPWL.

[3]Analytical tools generally model the wirelength as a quadratic term due to differentiability purposes. That is why those tools often rely on the star or clique models, where each net length is computed using the Euclidean distance (ALPERT; MEHTA; SAPATNEKAR, 2008).

2.1.4 Cell Density Control

Besides minimizing wirelength, placement tools must also spread the cells through the placement area to reduce cell density. Controlling the density is important to reduce cell overlaps and also to leave space for routing and for layout intrusive optimization techniques like gate sizing and buffer insertion (ALPERT; MEHTA; SAPATNEKAR, 2008; VISWANATHAN et al., 2010). Placement tools try to keep the density below a given limit. A common metric to estimate density is to divide the placement area into squares called bins and compute their utilization dividing the occupied area by the total bin area (excluding the bins occupied by macro blocks) (KIM et al., 2012). Figure 14 presents an example of how to divide the placement area into bins. An efficient metric used to measure the impact of placement density is the Average Bin Utilization (ABU), which will be detailed in Section 3.2.1.1.

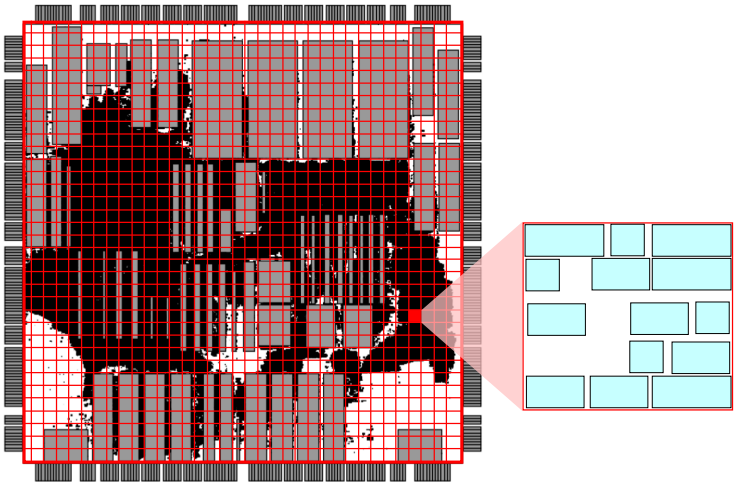


Figure 14: **Example of how to compute circuit cell density.** *To compute density, the placement area is divided into squares (bins) whose edges are 9 times larger than the row height. The utilization of each bin (excluding those occupied by macro blocks) is computed dividing the occupied area by the total area. Supposing that each bin has a total area of* $324\mu m^2$ *and the highlighted bin has an occupied area of* $250\mu m^2$*, thus the bin utilization is* $\frac{250}{324} \approx 0.77$*. Placement snapshot adapted from (SPINDLER, 2008).*

2.2 TIMING MODELING AND CONSTRAINTS

To evaluate the circuit performance and assess how far from the target clock frequency the circuit is, proper modeling of timing and adequate tracking of violations are required. This section firstly reviews the concepts of setup and hold timing constraints, which are both exploited in this thesis. Then it introduces timing analysis concepts and metrics through a small example. Finally, it details the adopted modeling for cell and interconnection electrical behavior.

2.2.1 Setup and Hold Timing Constraints

Timing constraints are applied to the circuit **combinational blocks**[4], delimited by the **timing startpoints** (output of sequential elements and circuit input pads) and the **timing endpoints** (input of sequential elements and circuit output pads) (SAPATNEKAR, 2004). The timing model captures setup and hold constraints separately by comparing them with late and early arrival times, respectively. This leads to two timing scenarios which are referred to as **late** and **early**. Figures 15 and 16 illustrate these scenarios by showing two registers, named R_1 and R_2, interleaved with a combinational block. To achieve a target frequency of $\frac{1}{P}Hz$, the clock period must be no greater than P time units.

The **setup time** corresponds to the amount of time during which a data signal must be stable at the input of a sequential element **before** the clock edge to ensure its correct sampling (capture) (SAPATNEKAR, 2004). The setup time of R_2 in Figure 15 is labeled as t_{su}. The sum of the maximum delay of R_1 (i.e. clock-to-output delay) and the maximum delay of the combinational block gives the **late arrival time** a^L at the input of R_2. The difference between P and t_{su} gives the latest time when data is allowed to be stable at the input of R_2 so as to respect the setup constraint, and is denoted as the **late required time** r^L. The **late slack** s^L is computed as the difference between r^L and a^L. A non-negative value indicates that the setup constraint is satisfied.

The **hold time** corresponds to the amount of time during which a data signal must be stable at the input of a sequential element **after** the clock edge to ensure its correct sampling (capture) (SAPAT-

[4] A combinational block implements a given logic that depends only on the block inputs and does not depend on any previous state.

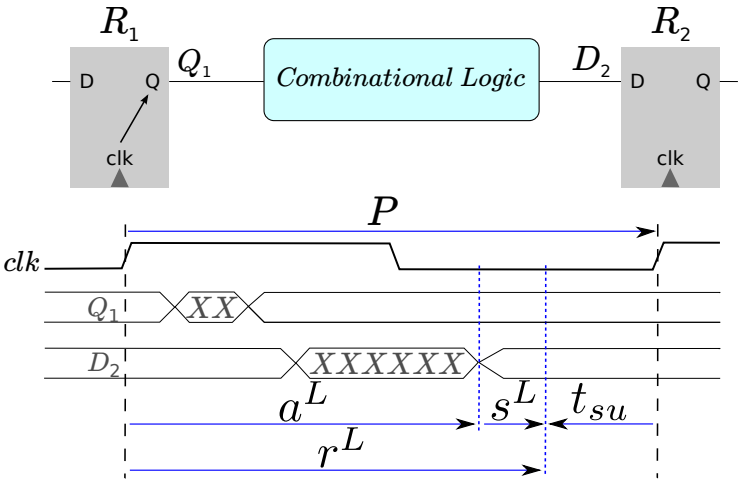


Figure 15: **Review of setup timing constraint (late scenario).**

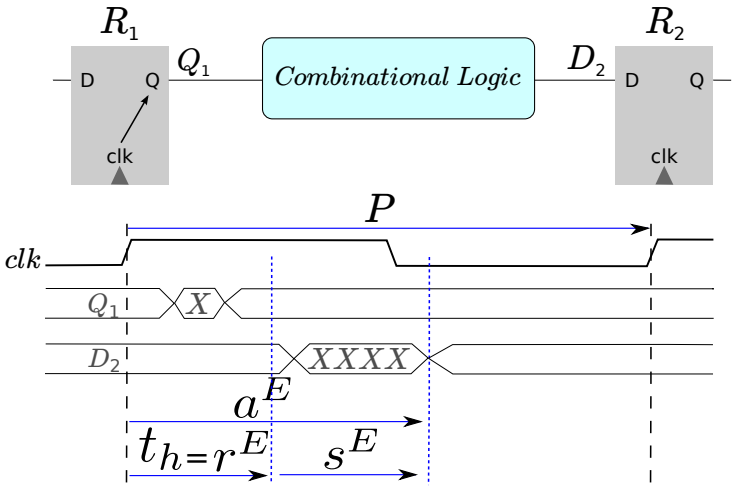


Figure 16: **Review of hold timing constraint (early scenario).**

NEKAR, 2004). The hold time of R_2 in Figure 16 is labeled as t_h. The sum of the minimum delay of R_1 and the minimum delay of the combinational block gives the **early arrival time** a^E at the input of R_2. The hold time t_h defines the earliest time when data is allowed to be stable at the input of R_2 so as to respect the hold constraint, and is denoted as the **early required time** r^E. The **early slack** s^E is computed as the difference between a^E and r^E. A non-negative value indicates that the hold constraint is satisfied.

2.2.2 Timing Analysis and Metrics

To propagate the maximum and minimum delays through each combinational block and compute the arrival times, required times, and slacks, a timing analysis engine must be employed. The most used technique to estimate the clock frequency of a given digital circuit is called static timing analysis (STA). The idea is to propagate the signals from timing startpoints to timing endpoints assuming pessimistic conditions for the late scenario and optimistic conditions for the early scenario[5] (BHASKER; CHADHA, 2009).

Timing analysis tools generally transform the hypergraph $G(V,\mathcal{N})$ into a directed acyclic graph (DAG) $D(V,E)$ by mapping each hyperedge $\mathcal{N}$ into a set of binary edges E. Then the DAG is partitioned into subgraphs representing each combinational block. Finally, the DAG is traversed in topological order[6] while arrival times are computed at each $v_j \in V$.

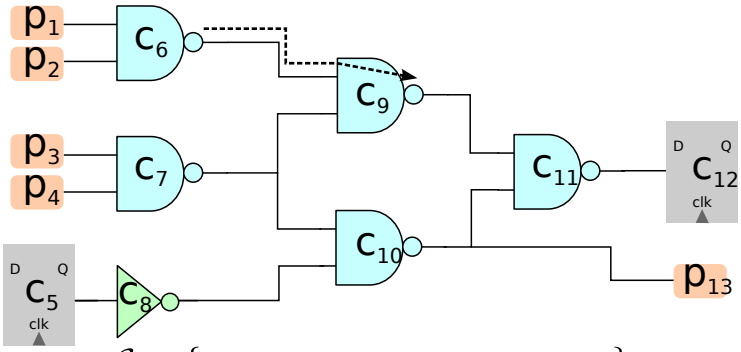
The boundaries of each combinational block are usually represented by two sets of integers, denoted as $\mathcal{TS}$ and $\mathcal{TE}$, which identify each vertex producing a timing startpoint or consuming a timing endpoint, respectively. The index of a vertex serves as its actual identifier. Formally, given a subgraph (V_k, E_k) with $V_k \subset V$ and $E_k \subset E$ representing a combinational block k, $\mathcal{TS}_k = \{i \mid i$ is integer and $v_i \in V_k$ produces a timing startpoint$\}$ and $\mathcal{TE}_k = \{i \mid i$ is integer and $v_i \in V_k$ consumes a timing endpoint$\}$. For simplicity, when referring to a given block, the index k is dropped in shorthand notation.

Timing analysis tools usually rely on the fanin and on the fanout of each vertex $v_j \in V$ to determine arrival times. Let $\mathcal{I}_j$ be the set of integers serving as identifiers for the vertices connected to its inputs, formally $\mathcal{I}_j = \{i \mid i$ is integer and $v_i \in V$ produces an input for $v_j \in V\}$. Let $\mathcal{O}_j$ represent the set of integers serving as identifiers for the vertices connected to its output, formally $\mathcal{O}_j = \{i \mid i$ is integer and $v_i \in V$ produces an output for $v_j \in V\}$. Figure 17 presents a small circuit to illustrate the aforementioned concepts.

Let $d^L_{i,j}$ and $d^E_{i,j}$ denote the late and early values for the delay measured between the output of cell c_i and the output of cell c_j (e.g. late and early delays from c_6's output and c_9's output (dashed arrow in Figure 17) are denoted $d^L_{6,9}$ and $d^E_{6,9}$, respectively). Given a combi-

[5] This is similar to the Critical Path Method used in project management (SAPATNEKAR, 2004).

[6] Corresponds to the ordering where a node v_j is processed only after all its predecessor nodes have been processed.



$$\mathcal{C} = \{c_5, c_6, c_7, c_8, c_9, c_{10}, c_{11}, c_{12}\}$$
$$\mathcal{TS} = \{1, 2, 3, 4, 5\} \qquad \mathcal{TE} = \{12, 13\}$$
$$\mathcal{I}_9 = \{6, 7\} \qquad \mathcal{O}_7 = \{9, 10\}$$

Figure 17: **Circuit example with timing analysis notation.** *A circuit example containing 5 timing startpoints, 8 standard cells, and 2 timing edpoints. The fanin set for c_9 and the fanout set for c_7 are also presented.*

national block, late and early arrival times of each vertex v_j, denoted as a_j^L and a_j^E, can be recursively defined from its timing startpoints towards its timing endpoints, as follows:

$$a_j^L = \max_{i \in \mathcal{I}_j}(a_i^L + d_{i,j}^L), \qquad a_j^E = \min_{i \in \mathcal{I}_j}(a_i^E + d_{i,j}^E) \tag{2.1}$$

The late (early) required time, denoted as r_j^L (r_j^E), corresponds to the latest (earliest) time when the signal transition must reach each timing endpoint to ensure the target clock frequency. Given a combinational block, the required times can be recursively defined from its timing endpoints towards its timing startpoints, as follows:

$$r_j^L = \min_{k \in \mathcal{O}_j}(r_k^L - d_{j,k}^L), \qquad r_j^E = \max_{k \in \mathcal{O}_j}(r_k^E - d_{j,k}^E) \tag{2.2}$$

To evaluate how far a design is from the target clock frequency, late and early slacks are tracked at timing endpoints (SINHA et al., 2013), as follows:

$$slk_j^L = r_j^L - a_j^L, \qquad slk_j^E = a_j^E - r_j^E, \ \forall j \in \mathcal{TE} \tag{2.3}$$

Figure 18 presents a small example of computing arrival times, required times, and slacks for both timing scenarios.

Timing optimization techniques such as incremental timing-driven placement and gate sizing, typically try to improve a timing metric known as worst negative slack (WNS). It can be defined, under late and early scenarios, as follows:

$$WNS^L = \min_{j \in \mathcal{TE}} (0, slk_j^L) \quad (2.4)$$

$$WNS^E = \min_{j \in \mathcal{TE}} (0, slk_j^E) \quad (2.5)$$

Note that WNS either captures the most severe violation (as a negative value corresponding to the timing endpoint with worst slack) or a non-violation (as a zero valued upper bound). For the example in Figure 18 (b), $WNS^L = -5$ and $WNS^E = 0$.

Although the WNS metric represents timing violations on the worst path, a circuit may have several other near-critical paths violating the timing constraints. That is a why a second metric, called total negative slack (TNS), captures the criticality of other critical paths, as follows:

$$TNS^L = \sum_{j \in \mathcal{TE}} \min(0, slk_j^L) \quad (2.6)$$

$$TNS^E = \sum_{j \in \mathcal{TE}} \min(0, slk_j^E) \quad (2.7)$$

Note that the TNS adds up the impact of all violations into a single number. For the example in Figure 18 (b), $TNS^L = -6$ and $TNS^E = 0$.

2.2.3 Cell and Interconnection Modeling

The timing analysis of a circuit requires the modeling of standard cell and interconnection electrical behaviors. Next, the adopted cell and interconnection modeling are explained.

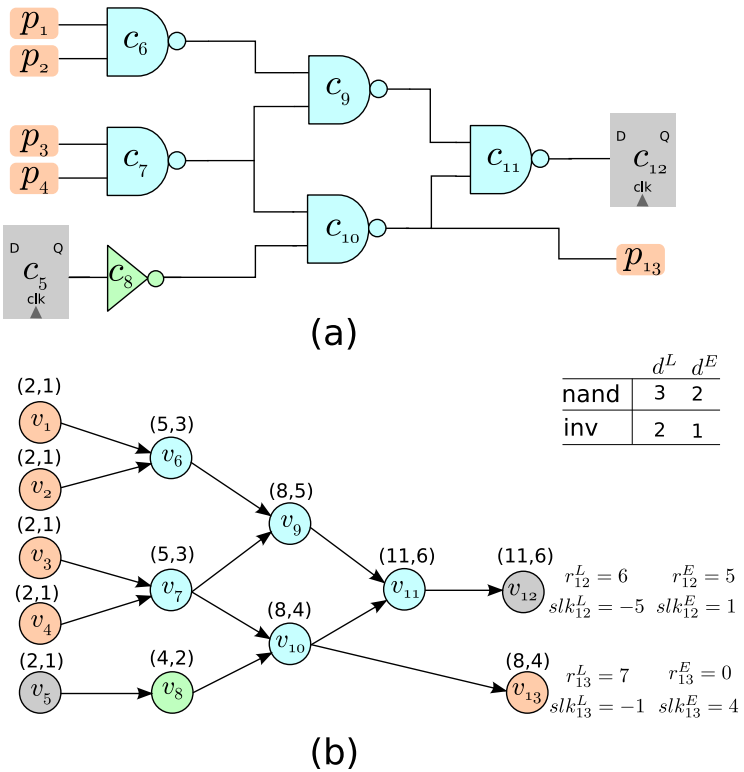


Figure 18: **Small example of static timing analysis.** *(a) A circuit example containing 5 timing startpoints, 6 combinational gates, and 2 timing endpoints. (b) The DAG for circuit example and timing analysis considering late and early scenarios. The small table on the right-hand side defines late and early delays for the combinational gates. The example assumes that interconnections delays are zero,* $t_{su} = 1, t_h = 5$ *and* $P = 7$*. The pairs* (a^L, a^E) *indicate the late and early arrival times obtained for each vertex. The arrival times for combinational gates are computed at their output pins. The example supposes that all timing startpoints have the same arrival times* $(2,1)$*. The resulting values for required times* (r^L *and* r^E) *and slacks* (slk^L *and* slk^E) *are shown for each timing endpoint. Since setup and hold constraints only apply to* v_{12} *(which represents a register) but not to* v_{13} *(which represents a pad), the late required time of the former is one time unit less than the latter and the early required time of the former is 5 whereas it is zero for the latter. Note that the paths ending at both vertices* v_{12} *and* v_{13} *violate the late timing constraints (negative slacks), although the early time constraints are satisfied (positive slacks).*

2.2.3.1 Cell Modeling and Concepts

The main electrical characteristics of cells are explained with the aid of Figure 19. Assume that the output of a cell c_j is wired to the inputs of other cells, say c_k and c_l. Since the timing of an output signal depends on which input signal transition has been observed, timing is defined between a given input, say i, and the output, say j. This notion, known as timing arc, is illustrated in Figure 19 (a). The following electrical characteristics are defined for each timing arc:

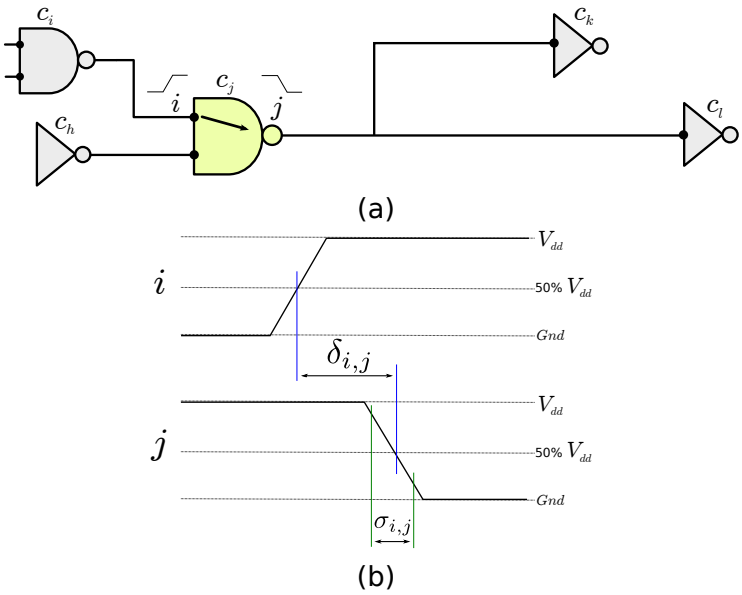


Figure 19: **Electrical characterization of a cell.** *(a) A subcircuit with emphasis on timing arc i,j of c_j. (b) Waveforms to illustrate the concepts of arc delay ($\delta_{i,j}$) and arc slew ($\sigma_{i,j}$).*

- **arc delay** of $c_j \in C$ with respect to its input i, denoted as $\delta_{i,j}$, corresponds to the time difference between the instant that the input signal crosses 50% of V_{dd} and the instant that signal j crosses 50% of V_{dd}. Although separate rise and fall arc delays are considered, such detailing is omitted for clarity.

- **arc slew** of $c_j \in C$ with respect to its input i, denoted as $\sigma_{i,j}$, corresponds to the time taken by the output signal j to complete a transition (fall or rise), measured between the voltages 80% of

V_{dd} and 20% of V_{dd}. Although separate rise and fall arc slews are considered, such detailing is omitted for clarity.

Static timing analysis tools account for the worst and the best arc slews over all timing arcs of a given cell, depending on whether the chosen timing scenario is late or early (BHASKER; CHADHA, 2009). The **worst arc slew** of $c_j \in C$, denoted as σ_j^L, is the maximum $\sigma_{i,j}^L$ over all inputs; the **best arc slew**, denoted σ_j^E, is the minimum $\sigma_{i,j}^E$.

During the design of standard cell libraries, each cell is characterized through electrical simulations so as to model each arc delay and arc slew as a function of cell downstream capacitance and cell input slew.

The delay and output slew of a cell have a direct correlation with the downstream capacitance (i.e., the larger the downstream capacitance, the longer the delay). Therefore, the downstream capacitance of a driver cell c_j, say C_j^{down}, must account for the effect of the interconnection capacitance and the effect of the input capacitance of fanout cells, say C_k^{in} and C_l^{in}. The adopted modeling for interconnection capacitance is described in Section 2.2.3.2.

Since delay and output slew of a cell also have a direct correlation with input slew (i.e., the longer the input slew, the longer the delay), the input slew at a given input pin i of c_j corresponds to the worst (best) arc slew of its driver[7] c_i in late (early) scenarios, denoted as σ_i^L (σ_i^E).

For technology nodes below 180 nm, the usual linear delay model no longer provides reasonable accuracy for the ranges of input slew and output capacitance (WESTE; HARRIS, 2010). Therefore, contemporary cell libraries adopt a Non-Linear Delay Model (NLDM) that stores arc delay and arc slew information of each cell in a bidimensional lookup table. Such industry standard format is known as *Liberty (.lib)* (BHASKER; CHADHA, 2009). These lookup tables store the pre-characterized information for various combinations of input slew and output capacitance, as exemplified in Table 1. When a given combination of input slew and output capacitance does not match to a lookup table entry, a linear interpolation is performed between the two closest entries. Therefore, late and early arc delay and arc slew values are obtained by queries to the library lookup tables, as in the Equations (2.8) and (2.9), where LUT_D and LUT_S correspond to the queries to delay and slew, respectively. Observe in these equations that C_j^{down} corresponds to a row index (i.e. Output Capacitance) in Table 1, while

[7] The slew degradation through the wire is detailed in Section 2.2.3.2.

Table 1: **Example of delay lookup table for a standard cell library.** *The delay information is accessed by mapping the corresponding output capacitance and input slew. For example, the mapping of input slew* 80.00 *and output capacitance* 4.00 *gives* 62.46*ps of delay).*

	Input Slew							
Output Capacitance	**5.00**	**30.00**	**50.00**	**80.00**	**140.00**	**200.00**	**300.00**	**500.00**
0.00	14.064	21.864	27.204	33.132	41.784	48.528	57.852	73.488
1.00	20.316	28.116	34.32	41.940	52.980	61.416	72.780	90.996
2.00	26.556	34.356	40.596	49.488	62.676	72.684	86.004	106.800
4.00	39.060	46.860	53.100	62.460	79.260	92.112	109.02	134.808
8.00	64.056	71.856	78.096	87.456	106.176	123.708	146.88	181.704
16.00	114.06	121.86	128.100	137.460	156.180	174.900	205.908	255.972
32.00	214.056	221.856	228.096	237.456	256.176	274.896	306.096	368.496

σ_i^L corresponds to a column index (i.e. Input Slew).

$$\delta_{i,j}^L = LUT\_D(C_j^{down}, \sigma_i^L) \qquad \delta_{i,j}^E = LUT\_D(C_j^{down}, \sigma_i^E) \quad (2.8)$$

$$\sigma_{i,j}^L = LUT\_S(C_j^{down}, \sigma_i^L) \qquad \sigma_{i,j}^E = LUT\_S(C_j^{down}, \sigma_i^E) \quad (2.9)$$

2.2.3.2 Interconnection Modeling and Concepts

The main concepts involved in the electrical characterization of an interconnection are explained using Figure 20. Assume that the output of a cell c_i is wired to the inputs of other cells, say c_k and c_j.

The distributed resistance and capacitance (RC) of an interconnection are modeled as several lumped RC segments. Among the lumped models available (L-model, T-model, and π-model), the present work employs the π-model because it provides the best accuracy and is fast to simulate (as it produces less RC nodes when compared to the T-model) (WESTE; HARRIS, 2010). Therefore, each interconnection is modeled as an RC tree with a set of segments $\mathcal{S}$, wherein each π-segment has one resistor (representing the lumped segment resistance) and two capacitors (each capacitor containing half of the segment capacitance). Figure 20 (b) shows an RC tree with four π-segments. Observe that the second capacitor (from left to right) lumps half of the capacitance of segments 1, 2, and 3. Also observe that the pin capacitance for the net receivers (i.e. c_k and c_j) are denoted C_k^{in} and C_j^{in}.

The impact of a given interconnection in the neighboring subcircuit is mainly due to 3 factors:

1. **Driver's downstream capacitance**: The capacitance to be

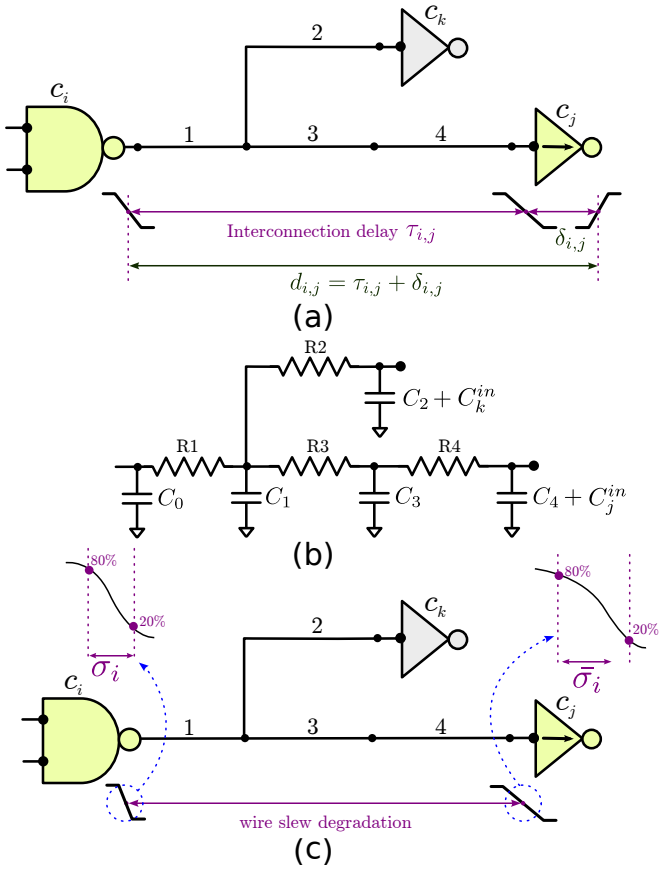


Figure 20: **Electrical characterization of an interconnection.** *(a) A subcircuit to illustrate the Elmore delay from the output of c_i to the input of c_j ($\tau_{i,j}$). The overall delay of a cell c_j ($d_{i,j}$) corresponds to the sum of $\tau_{i,j}$ and the respective arc delay $\delta_{i,j}$. (b) Interconnection modeled using the lumped π-model. (c) The original slew at the output of cell c_i (σ_i) and its degradation at the input of cell c_j ($\bar{\sigma}_i$). Observe a steeper transition at the input of c_j when compared to the transition at the output of c_i, indicating a degradation of the signal slew.*

charged by the interconnect driver (e.g. c_i) affects its arc delays and slews, as already detailed in Section 2.2.3.1. The present work estimates the downstream capacitance of a driver cell using the lumped capacitance model, which combines all the interconnect capacitances into a single capacitor (RABAEY; CHANDRAKASAN; NIKOLIC, 2003). For instance, considering the

RC network in Figure 20, the estimate for $C_i^{down} = C_0 + C_1 + C_2 + C_3 + C_4 + C_k^{in} + C_j^{in}$. This model is simple enough to be used in early stages of the design flow, but has a few drawbacks, especially for late stages, as it does not account for the resistive shielding effect. In late stages of the design flow, higher order models can be used to estimate the interconnect effective capacitance (KASHYAP et al., 2002).

2. **Interconnection delay**: The delay of a given interconnection can be estimated using the well-known Elmore delay model (ELMORE, 1948). Basically, it models the wire delay between two points (say, c_i and c_j in Figure 20) as the first moment of the impulse response (ELMORE, 1948). The Elmore delay of a given segment $s_i \in \mathcal{S}$ can be computed as $R_i \cdot C_i^{down}$, where R_i corresponds to the segment resistance and C_i^{down} corresponds to the segment downstream capacitance, i.e. the capacitance to be charged by segment i (e.g. in Figure 20, $C_3^{down} = C_3 + C_4 + C_j^{in}$). Therefore, the delay between two points (say, c_i and c_j) can be computed as the summation of the Elmore delay at each segment in the path (e.g. segments 1, 3, and 4), as detailed in Equation (2.10). The Elmore delay model is widely adopted in the literature since it has a closed formula that requires only resistances and capacitances, and is proved to be an upper bound for the real delay for any RC tree (ALPERT; MEHTA; SAPATNEKAR, 2008). In addition, it also provides important properties to be used in optimization algorithms, such as convexity. That is why it has been widely used, especially in initial stages of the design flow. Nevertheless, higher-order models must be used to improve the accuracy of the Elmore delay during late stages like Asymptotic Waveform Evaluation (PILLAGE; ROHRER, 1990).

$$\tau_{i,j} = \sum_{s_k \in path(i \to j)} R_k C_k^{down} \tag{2.10}$$

3. **Interconnection slew**: Up to this point, we have assumed that the input slew of a cell c_j with respect to an input i was the worst or the best slew of a cell c_i, depending on the chosen timing scenario. However, the input slew degrades along the interconnection, as illustrated in Figure 20 (b). In this work, the wire slew degradation is computed using the PERI model (KASHYAP et al., 2002). Equations (2.11) and (2.12) present the slew degra-

dations for each timing scenario slew, where $\beta_{i,j}$ is the second moment of the impulse response[8] for cell c_j as detailed below.

$$\bar{\sigma_i^L} = \sqrt{(\sigma_i^L)^2 + (2\beta_{i,j} - \tau_{i,j})} \quad (2.11)$$

$$\bar{\sigma_i^E} = \sqrt{(\sigma_i^E)^2 + (2\beta_{i,j} - \tau_{i,j})} \quad (2.12)$$

The overall delay of a given cell c_j with respect to its input i, denoted as $d_{i,j}$, is measured between the output of its driving cell c_i and the output of c_j itself, as illustrated in Figure 20 (a). Thus, the overall delay is a sum of two contributions: the contribution of interconnection delay $\tau_{i,j}$ and the contribution of arc delay $\delta_{i,j}$. Therefore, the overall delays of c_j in the late and early scenarios are:

$$d_{i,j}^L = \tau_{i,j} + \delta_{i,j}^L \quad (2.13)$$

$$d_{i,j}^E = \tau_{i,j} + \delta_{i,j}^E \quad (2.14)$$

It is worth mentioning that the delay models employed inside optimization engines are different from those used by industrial timing analyzers. The approximative interconnect models presented in this section (like Elmore's delay) are generally more adequate for optimization engines, because they can be denoted by closed formulas and have important properties (like convexity). On the other hand, industrial timing analysis tools must employ high-order models (some not denotable by closed formulas) to ensure accurate timing estimates during the physical synthesis flow (OZDAL; BURNS; HU, 2012).

2.3 REVIEW OF LAGRANGIAN RELAXATION

This section reviews the basic concepts of Lagrangian Relaxation for a clear understanding of this thesis[9].

Lagrangian Relaxation (LR) is an efficient technique to optimize constrained problems, as in the case of the problems to be tackled in the thesis. Some advantages of LR are scalability and flexibility, allowing to

[8] The second moment of the impulse response is computed similarly to the Elmore delay, but multiplying each RC delay by the first moment of Elmore delay as: $\sum_{k \in M_{i,j}} (R_k \cdot C_k^{down} \cdot \tau_{i,j})$.

[9] This section reproduces (in English) part of the Section 2.5 from the author's Master Thesis (LIVRAMENTO, 2013), written in Portuguese.

optimize large problems of different nature such as linear, convex, non-linear, combinatorial etc[10]. The basic idea of LR is to convert a problem with hard constraints into an easier-to-solve problem by removing the hard constraints from the original problem and incorporating them into the objective function of the new problem[11]. Each relaxed constraint in the new objective function is multiplied by a penalty term called Lagrangian Multipler (LM) (FISHER, 1985) (BOYD; VANDENBERGHE, 2004).

The basic concepts of Lagrangian Relaxation are explained through its application to the constrained shortest path problem, which is an NP-hard problem (AHUJA; MAGNANTI; ORLIN, 1993). Assume a directed graph $D(V, E)$ with a source node S and a terminal node T. Each edge $e \in E$ represents a path with two associated values: the cost and the time to traverse the edge, denoted by c_e and t_e, respectively. Figure 21 (a) illustrates this definition. The constrained shortest path problem has to find a path from S to T with the lowest cost, whose traversal time is lower than or equal to the constraint b [12].

The mathematical formulation as an integer linear problem is presented through Equations 2.15 to 2.18, called Primal Problem (PP). Equation 2.15 defines the objective function while Equation 2.16 guarantees the existence of a path between S and T. Finally, Equation 2.17 provides a constraint b for the traversal time between S and T, while Equation 2.18 restrics x_e to a binary variable.

[10] For more details of linear and non-linear problems, refer to (BAZARAA; SHERALI; SHETTY, 2006). A more extensive literature about convex problems can be found in (BOYD; VANDENBERGHE, 2004). A good reference for combinatorial problems is (KREHER; STINSON, 1999).

[11] There is no rule of thumb to identify which constraints should be removed or not, generally being chosen in an empiric way (FISHER, 1985). More details and examples of identifying the constraints to be removed can be found in the remaining of this section.

[12] This illustrative example uses a single constraint for simplicity, albeit problems tackled in the Chapters 3 and 4 contain many more constraints.

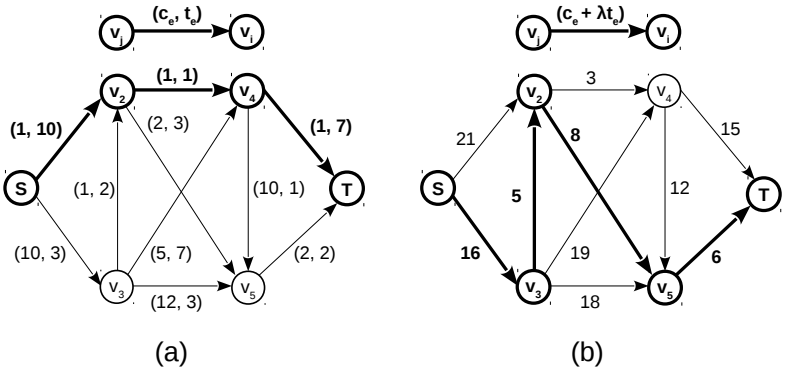


Figure 21: **Example of application of Lagrangian Relaxation.** *(a) An instance of the constrained shortest path problem. Each edge $e \in E$ has an associated pair of values (c_e, t_e), where c_e and t_e represent the cost and time needed to traverse the edge e, respectively. (b) Example of the application of LR into the shortest path problem, assuming $\lambda = 2$. Observe that the new associated cost of each edge is the value $c_e + \lambda t_e$. Adapted from (AHUJA; MAGNANTI; ORLIN, 1993)*

.

$$\textbf{Min.} \quad : \quad \sum_{e \in E} c_e x_e \tag{2.15}$$

$$\textbf{S. t.} \quad : \quad \sum_{e \ enters \ v_i} x_e \; - \sum_{e \ leaves \ v_i} x_e = \begin{cases} -1 & \text{if } v_i = S \\ +1 & \text{if } v_i = T \\ 0 & \text{otherwise} \end{cases} \tag{2.16}$$

$$\sum_{e \in E} t_e x_e \leq b \tag{2.17}$$

$$x_e \in \{0,1\}, \forall e \in E \tag{2.18}$$

Suppose that the problem constraints can be divided in two different sets. The constraint defined in Equation 2.17 is hard to handle during the problem optimization and its removal turns the PP into an easier-to-solve problem, while the remaining constraint can be easily handled. Note that if the constraint defined in Equation 2.17 is removed, the PP is converted into the classical shortest path problem. The basic idea consists in removing the constraint defined in Equation 2.17 and incorporating them into the new objective function. Such

constraint is multiplied by the penalty term Lagrange Multiplier (λ), giving rise to the Lagrangian Function (LF) defined in Equation 2.19.

$$L_\lambda(x) = \sum_{e \in E} c_e x_e + \lambda(\sum_{e \in E} t_e x_e - b) \quad (2.19)$$

The problem is now addressed by iteratively solving two subproblems:

1. The **Lagrangian Relaxed Subproblem (LRS)**, formalized through Equations 2.20 a 2.22, has to minimize the LF defined in Equation 2.19. Since the new problem corresponds to the classic shortest path problem, the optimal solution of LRS can be found in polynomial time using consolidated algorithms in the literature, such as *Dijkstra* or *Bellman-Ford* (CORMEN et al., 2009). Figure 21 (b) illustrates the cost associated to each edge after incorporating the maximum traversal time constraint into the objective function. It is worth mentioning that the penalty term (λ) assumes a fixed non-negative value during the resolution of LRS. An important property from LRS is that its optimal solution $z(\lambda)$ for any $\lambda \geq 0$ leads to a lower bound $z(\lambda) \leq z^*$, where z^* corresponds to the optimal solution of the PP. It is straightforward to verify such property, since any feasible solution for the PP defines a path from S to T, such that $\sum_{e \in E} t_e x_e \leq b$. Therefore, any feasible solution for the PP is also a feasible solution for the LRS and, as the LM only assumes non-negative values, the following property is verified $\sum_{e \in E} c_e x_e + \lambda(\sum_{e \in E} t_e x_e - b) \leq \sum_{e \in E} c_e x_e$. As consequence, $z(\lambda) \leq z^*$ (BOYD; VANDENBERGHE, 2004).

$$\textbf{Min.} \quad : \quad L_\lambda(x) \quad (2.20)$$

$$\textbf{S. t.} \quad : \quad \sum_{e \ enters \ v_i} x_e - \sum_{e \ leaves \ v_i} x_e = \begin{cases} -1 & \text{if } v_i = S \\ +1 & \text{if } v_i = T \\ 0 & \text{otherwise} \end{cases} \quad (2.21)$$

$$x_e \in \{0,1\}, \forall e \in E \quad (2.22)$$

2. Since $z(\lambda)$ defines a lower bound for z^*, the **Lagrangian Dual Problem (LDP)** has to obtain a tighter lower bound for z^*, as formalized in Equations 2.23 and 2.24. The objective of LDP

is to maximize the solution found during LRS ($z(\lambda)$), by updating the LMs (λ) accordingly. A widely used method to solve the LDP for non-differentiable functions (as in the case of the formulation considered) is the subgradient method (FISHER, 1985). The basic idea of the subgradient method is to update the Lagrange Multipliers based on the subgradient (name of the term $\sum_{e \in E} t_e x_e - b$) that indicates the direction in which the LMs must increase or decrease, according to the violation (or not) of the constraint b. Therefore, the LM of a given iteration $k+1$ can updated through the subgradient method as: $\lambda_{k+1} = max\{0, \lambda_k + \rho_k(\sum_{e \in E} t_e x_e - b)\}$, where ρ_{k+1} corresponds to the step size and must be a non-negative value (generally defined empirically)[13].

$$\textbf{Max.} \quad : \quad z(\lambda) \tag{2.23}$$

$$\lambda \geq 0 \tag{2.24}$$

Under a specific condition known as strong duality the optimal value of LDP is also the optimal value of PP (i.e., $z(\lambda) = z^*$), implying that the optimal duality gap is zero. Unfortunately, the problems to be tackled in this thesis are NP-Complete and therefore strong duality does not hold (i.e., duality gap is positive). Indeed, most physical design problems are NP-complete and Lagrangian Relaxation is generally applied as a fast technique to efficiently explore the search space, but is generally complemented with other techniques (OZDAL; BURNS; HU, 2012) (LI et al., 2012) (KIM et al., 2012) (LIVRAMENTO et al., 2014). More details on applying Lagrangian Relaxation to timing optimization problems are presented in Chapters 3 and 4.

[13] The subgradient method always converge to the optimal solution if the sequence of steps satisfies the following conditions: $\lim_{k \to \infty} \rho_k = 0$ e $\sum_{k=1}^{\infty} \rho_k = \infty$ (FISHER, 1985).

3 INCREMENTAL TIMING-DRIVEN PLACEMENT

Timing closure is a critical task for modern SoC design (KAHNG, 2015). Although, in earlier technology generations, the delays of logic gates were dominant over interconnection delays, the thinner and more resistive wires in nanometer technologies became the bottleneck for circuit timing (SHELAR; PATYRA, 2010; SAXENA et al., 2004), making cell locations crucial to improving circuit delays. For this reason, timing-driven placement (TDP) became a crucial step towards timing closure (ALPERT; CHU; VILLARRUBIA, 2007; ALPERT et al., 2012). Unfortunately, most TDP techniques focus on signal nets and overlook the impact of register location on clock tree quality, in spite of the fact that a poor register placement may increase clock wire capacitance and largely affect power consumption. Indeed, 30%–70% of an SoC power consumption comes from the clock tree (LEE; KIM; MARKOV, 2010; GUTHAUS; WILKE; REIS, 2013). Since a clock tree is essentially defined by the location of the sequential elements, register placement plays a key role towards a compact clock tree. However, many register placement techniques found in the literature disregard its impact on timing and routability.

Since register placement largely affects both clock tree power and routability metrics like signal wirelength and density, TDP and register placement should be properly coupled for a successful synthesis. However, to the best of author's knowledge, their coupling is not reported in the literature. For an effective coupling, the optimization obtained from one technique should not undermine the other's. If register placement was applied after TDP, the wiring between sequential and combinational elements would be largely touched. However, when register placement is performed beforehand, only the registers in critical paths may be touched by TDP and they represent a small percentage of the total number of registers (which, in turn, corresponds to around 15% of the standard cells (CHEON et al., 2005; LEE; MARKOV, 2012).

The reasons above motivated us to investigate how to properly couple register placement and TDP and how to design the underlying techniques accordingly. This chapter reports the main contributions to this subject:

1. A new approach that handles sequential and combinational elements separately, but relies on two keys for an effective coupling: First, only the registers in critical paths may be touched by TDP; Second, the shortening of clock wirelength is obtained with lim-

ited variation in signal wirelength and placement density. The approach consists in first applying incremental register placement (guided by a virtual clock tree to reduce clock wiring capacitance while preserving global placement quality) and then performing incremental TDP to minimize the total negative slack.

2. A novel technique for incremental register placement (IRP) that targets clock-tree compactness with bounded impact on routability. It combines clock-net contraction and register clustering forces so as to reduce the clock wirelength.

3. A new Lagrangian Relaxation formulation for TDP that minimizes the total negative slack for both setup and hold timing violations, where Lagrange multipliers are used as net weights and are dynamically updated using an accurate timing analyzer. To solve the formulation, this work proposes a technique that relies on a novel discrete search and employs Euclidean distance to define a proper neighborhood.

The remaining of this chapter is organized as follows. Section 3.1 first reviews related works on IRP and TDP and then discusses the limitations that preclude their proper coupling. Section 3.2 formalizes the target problem and its decomposition. Section 3.3 describes the proposed approach and the algorithms employed to solve the target problem. Section 3.4 shows the experimental evaluation of the proposed approach. Finally, Section 3.5 draws the main conclusions.

3.1 RELATED WORKS

This section first reviews related works on incremental timing-driven placement and incremental register placement. Then it discusses the hurdles to overcome for their proper coupling.

3.1.1 Works on Incremental Timing-Driven Placement

TDP techniques can be classified into net-based, path-based, or hybrid.

Net-based techniques translate timing information into net-weights or net-length constraints and minimize a weighted wirelength objective. The basic idea of net-weighting techniques is to assign higher weights to critical nets so as to guide the placement engine towards

shorter nets. Net-length techniques specify constraints on critical nets so as to bound their maximum lengths. Net-based techniques can be either static or dynamic.

Static approaches generate net-weights that are kept unchanged during optimization. **Kong (2002)** proposed a path counting scheme so as to assign net-weights according to the number of critical paths passing through the nets, also known as path sharing. **Ren, Pan and Kung (2005)** presented a sensitivity-based technique to predict the impact of net-weights on worst and total negative slacks. They also experimentally showed the importance of minimizing total negative slack during TDP to achieve timing closure. The main drawback of static approaches is to rely on a single step to generate the net-weights, which may become inaccurate during the optimization process.

Dynamic approaches iteratively update net-weights. They rely on a timing analysis engine to keep an up-to-date timing profile. **Eisenmann and Johannes (1998)** devised an iterative net-weighting strategy that was embedded into a force-directed placer. Such strategy relied on historic information to avoid drastic oscillation of the net-weights during the optimization. **Riess and Ettelt (1995)** proposed the use of a star interconnection model to accurately compute the net delay using Elmore's model and minimized a weighted quadratic objective function. To reduce oscillation, the net-weights are dynamically updated using information from the two previous iterations. **Kahng and Wang (2004)** employed an iterative conjugate gradient method where the weights of critical nets are increased using a criticality exponent to emphasize their higher importance over non-critical nets. **Halpin, Chen and Sehgal (2001)** proposed a linear programming formulation to generate net-length constraints using the Half-Perimeter Wirelength (HPWL) net model. Then an analytic placement, based on the bi-section method, is used to shorten the critical nets and satisfy the net-length constraints. **Rajagopal et al. (2003)** employed a force-directed placement using additional forces to meet the net-length constraints on critical nets. The proposed method allowed to specify net-length constraints for each interconnection segment separately. The main limitation of such dynamic approaches is to ignore macro blocks and overlaps during optimization. This may lead to severe timing degradation during the legalization step.

Path-based approaches minimize timing violations through accurate modeling of the timing of circuit paths, generally using linear programming (LP). **Srinivasan, Chaudhary and Kuh (1991)** and **Hamada, Cheng and Chau (1993)** proposed the use of primal-

dual approaches based on Lagrangian Relaxation to convert a constrained LP formulation into an unconstrained problem. Then such unconstrained formulation was solved to iteratively minimize the delay of critical paths. **Swartz and Sechen (1995)** employed simulated annealing to improve the timing of critical paths. **Chowdhary et al. (2005)** devised an LP formulation that models the circuit timing through the variation of delay and slew with respect to an accurate timing analyzer. This mechanism, called differential timing analysis, was later improved by **Ren et al. (2007)**, which proposed a pin-based timing LP formulation to reduce wirelength without timing degradation. **Moffitt et al. (2008)** presented a discretized approach that uses branch-and-bound to choose the location of cells. The main drawback of path-based approaches is scalability in face of large circuits, due to the exponential number of paths to be optimized. Another drawback is to ignore macro blocks and overlaps while solving the LP formulations.

Hybrid TDP techniques combine features of net and path-based approaches. **Luo, Newmark and Pan (2006)** proposed a path delay sensitivity function to generate net-weights and then minimized the wirelength of critical nets using LP. The technique is mainly net-based, but takes advantage of the path delay sensitivity to simultaneously optimize the critical path and a few logically adjacent paths, which is called *criticality adjacency network*. **Viswanathan et al. (2010)** presented an iterative approach that relies on an accurate timer to smooth the circuit critical paths without disrupting the wirelength. Its framework periodically performs slack histogram compression using additional techniques, such as buffer insertion and gate sizing, to recover the timing of non-critical paths. The authors claim that this improves the algorithm convergence.

Recently, **Bock et al. (2015)** proposed a local search algorithm to improve the worst negative slack of a circuit relying on a composition of three strategies: 1) a local search mechanism (through a supergradient ascent) tries to find a move direction (up, down, left and right) such that the local slack is improved, 2) a path-straightening algorithm performs a line search over the Euclidean distance between critical cells, and 3) neighboring cells with similar slack are clustered and moved together. According to the authors, most of the improvement comes from moving clustered cells.

Notice that most TDP techniques disregard the total negative slack and focus on the worst negative slack. Although the work from **Ren, Pan and Kung (2005)** proposed a net-weighting scheme to minimize a total negative slack function, the net-weights are generated

a priori and remain constant during optimization, which can undermine timing optimization. **Viswanathan et al. (2010)** mainly focus on minimizing the worst negative slack while the total negative slack is minimized in a few iterations. Besides, none of the previous works considered hold timing violations.

3.1.2 Works on Incremental Register Placement

Among the works on incremental register placement, the following review focuses on those relying on a simplified clock tree structure to guide the placement of sequential elements, which can be generated statically or dynamically. Static generation creates a virtual clock tree based on the initial register locations, which is kept unchanged during the whole optimization process. Dynamic generation iteratively updates the virtual clock tree each time sequential elements are relocated.

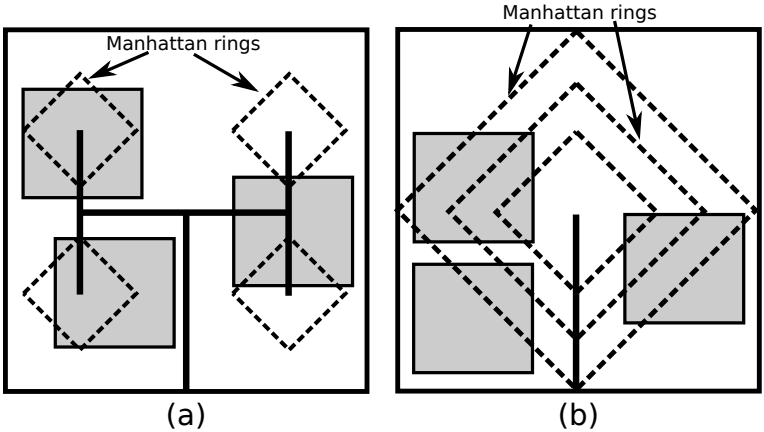


Figure 22: **Manhattan rings overlapping with macro blocks (gray)**. *Manhattan rings are identified by dashed lines while macros are represented by gray rectangles. (a) Manhattan rings driven by an H-tree. Note that the boundaries of the Manhattan rings (dotted) have the same distance from the clock source, located in the bottom of the circuit. (b) The boundaries of each Manhattan ring have a different distance from the clock source, allowing to employ different skew targets. Adapted from (LU et al., 2005) and (LEE; MARKOV, 2012).*

A few related works targeting zero clock skew rely on static structures called **Manhattan rings** (LU et al., 2005; HUANG et al., 2005;

LU et al., 2005). This notion is illustrated in Figure 22 (a). The idea consists in adding pseudo pins, located on the boundaries of Manhattan rings, and pseudo nets, connected to registers, so as to iteratively move registers towards zero skew. The use of several Manhattan rings for non-zero skew targets was also addressed in these works, as illustrated in Figure 22 (b). The main limitation of using Manhattan rings for register placement is their poor guidance in the presence of macro blocks, as also exemplified in Figure 22. Since modern SoCs are composed of several macro blocks, the use of regular shaped clock networks, such as Manhattan rings should be precluded (LEE; MARKOV, 2012).

Another class of related works uses implicit clock trees derived from the **bounding boxes** of register locations to shorten the clock wirelength (CHEON et al., 2005; WANG et al., 2007; PAPA et al., 2011). Through a motivational experiment, **Cheon et al. (2005)** showed that most of the clock network capacitance is at leaf level (including the wire capacitance and the input capacitance of sequential elements). Their algorithm identifies clusters of registers, based on the Manhattan distance between registers, as a first optimization step. Then a net-weighting technique is applied to shrink the bounding boxes of those clusters and to reduce the wire capacitance. **Wang et al. (2007)** devised a method for constructing a dynamic virtual clock tree to guide register placement. The virtual clock tree is then integrated into a force-directed placer that, through attractive forces between registers and between sibling nodes, aims to shorten the clock network, as illustrated in Figure 23. They also considered the use of additional forces during the contraction of the bounding boxes to prevent excessive increase of signal nets. **Papa et al. (2011)** presented the industrial methodology for clock tree synthesis of high-performance designs adopted in IBM. They employed a technique based on K-means[1] to generate the clusters of registers. Their strategy also relied on bounding boxes to reduce the size of clusters. The main drawback of this class of approaches is the use of simplified clock trees derived from the bounding boxes of register locations. According to Lee and Markov (2012), such use does not lead to an effective reduction in the final clock tree. In addition, all works employing bounding boxes ignore the presence of macro blocks.

Instead of using a simplified structure to guide register placement, **Lee and Markov (2012)** adopted a clock router to generate an **accurate dynamic virtual clock tree** to steer register placement

[1]K-means is a popular clustering technique that aims to partition a set of observed data into k sets (clusters).

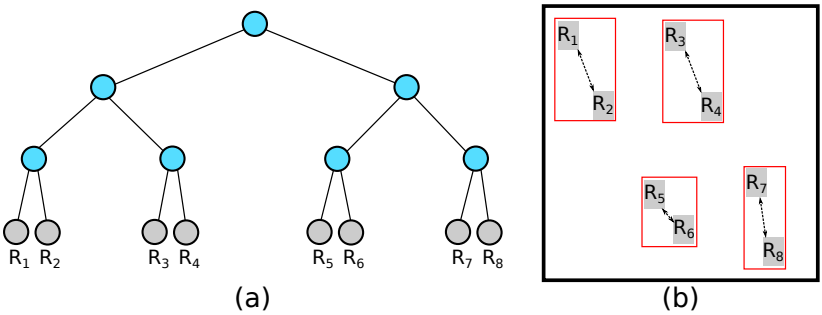


Figure 23: **Examples of bounding boxes to guide register placement.** *(a) A local branch of a clock-tree containing eight registers. (b) The idea is to shorten the clock net at the leaf level by reducing the size of bounding boxes (red). To reduce the leaf bounding boxes, attractive forces (dashed arrows) are used between register. Adapted from (WANG et al., 2007).*

towards power reduction. The authors employed a deferred merge embedding algorithm to generate a zero-skew clock tree that considers macro blocks. One of their contributions was to replace the forces between siblings in Wang et al. (2007) by attractive forces between parent and child nodes of the virtual clock tree, which were called arboreal clock-net contraction forces. According to the authors, this leads to a shorter clock tree as compared to the bounding box approach. Although the use of clock-net contraction forces leads to a shorter clock tree, they may be not effective to reduce clock tree capacitance at leaf level, which corresponds to 80% of the total capacitance (CHEON et al., 2005; GUTHAUS; WILKE; REIS, 2013). Another limitation is the lack of a mechanism to prevent excessive increase in the placement density profile during register placement.

3.1.3 The Hurdles Blocking a Proper Coupling

Most works on TDP found in the literature place sequential and combinational cells indistinguishably, overlooking the impact on the clock tree. Although Luo et al. (2008) and Papa et al. (2008) proposed techniques to account for the relocation of imbalanced registers (positive slack on one side and negative slack on the other side), the integration with register placement and its impact on clock tree wirelength was not addressed. On the other hand, incremental register placement techniques mainly focus on reducing clock tree wirelength, but disre-

gard timing and density metrics. Indeed, the percentage of sequential elements among all standard cells is around 15% on average (CHEON et al., 2005; LEE; MARKOV, 2012). As consequence, the incremental register placement step may drastically affect both circuit routability (increasing signal wirelength and density) and timing (due to clock skew and wiring between sequential and combinational elements). We observed that related techniques either perform incremental register placement or incremental timing-driven placement. Therefore, a proper integration between IRP and ITDP is needed. However, to the best of author's knowledge no related work addressing these issues was reported in the literature. Since register placement largely affects both clock tree synthesis and timing closure, it should be properly coupled with TDP so that the optimization obtained from one technique does not undermine the other's, avoiding disruptions on the quality of solution (e.g. signal wirelength and density). The work from Papa et al. (2011) proposed the co-design of register placement and clock network synthesis for high-performance circuits. The authors identified limitations in the IBM physical synthesis flow and proposed a reordering of optimization steps. They performed register clustering and then employed net-weights to reduce their bounding boxes. Although they looked into the impact of register placement on the quality of solution, they neither addressed clock wirelength nor placement density. In addition, no proper coupling with incremental timing-driven placement was addressed. The reasons above motivated the investigation of how to properly couple IRP and ITDP in the frame of a more general optimization problem. The next section formulates such a joint target problem.

3.2 PROPOSED PROBLEM FORMULATION

This chapter formulates the target optimization problem and its decomposition into subproblems. Then it shows how to tailor instances of such subproblems so as to avoid disruptions that could impair the quality of the solution of the target problem via decomposition. Finally, it shows how one of the instances can be cast into a Lagrangian Relaxation formulation.

Although the basics of the adopted design representation were already introduced in Chapter 2, some concepts are reproduced here for clarity. Besides, a more detailed representation is also introduced to properly formulate the target problem, as follows.

A sequential circuit can be represented by a set $\mathcal{C}$ of standard cells, a set $\mathcal{TS}$ of timing startpoints and a set $\mathcal{TE}$ of timing endpoints (REN et al., 2007). The set $\mathcal{TS}$ includes **both** circuit input pads and registers output pins. The set $\mathcal{TE}$ includes **both** circuit output pads and register input pins (OZDAL; BURNS; HU, 2012). There is also a set $\mathcal{N}$ of nets representing the interconnections between these elements. The set $\mathcal{C}$ can be partitioned into subsets $\mathcal{S}$ and $\mathcal{NS}$ of sequential and non-sequential elements, respectively, i.e. $\mathcal{C} = \mathcal{S} \cup \mathcal{NS}$ and $\mathcal{S} \cap \mathcal{NS} = \emptyset$. Each cell $c_j \in \mathcal{C}$ occupies a location $(x_j, y_j) \in \mathbb{Z}^+ \times \mathbb{Z}^+$ in the circuit layout. A circuit placement is a mapping $P : \mathcal{C} \mapsto \mathbb{Z}^+ \times \mathbb{Z}^+$ such that $P(c_j) = (x_j, y_j)$. The target problem can be formulated as follows.

Clock-tree-aware incremental timing-driven placement: Given a placement P, find a new placement P^* that minimizes clock tree capacitance and the number of timing violations.

This work proposes an approach to solve the target problem by solving two subproblems, which can be formulated as follows.

Incremental register placement (IRP): Given a placement P, find a mapping $R : \mathcal{S} \mapsto \mathbb{Z}^+ \times \mathbb{Z}^+$ that minimizes clock tree capacitance and induces a new placement P' such that $P'(c_j) = R(c_j)$ for each $c_j \in S$.

Incremental timing-driven placement (ITDP): Given a placement P', find a new placement P^* that minimizes the number of timing violations.

3.2.1 Tailoring Problem Instances for an Efficient Coupling

To properly address the target problem via decomposition, this work proposes to tailor instances of the subproblems by appropriate choices of constraints and objective functions.

3.2.1.1 Choice of Constraints

To make sure that both subproblems find legal placements, the constraints encoded through Equations (3.1) to (3.5) are defined for every $c_j \in \mathcal{C}$. Chip physical dimensions are denoted as X_{left}, X_{right}, Y_{bottom}, Y_{top}, W_{site}, and H_{row}; standard cell layout dimensions are denoted as W_j and H_j. Equations (3.1) and (3.2) ensure that the chosen location (x_j, y_j) for every cell c_j is within chip boundaries. Equations (3.3) and (3.4) ensure that every c_j is aligned to a standard cell site

and to a row, respectively. Equation (3.5) makes sure that cells do not overlap in a given row.

$$X_{left} \leq x_j \leq X_{right} - W_j \tag{3.1}$$
$$Y_{bottom} \leq y_j \leq Y_{top} - H_j \tag{3.2}$$
$$x_j = w \times W_{site},\ w \in \mathbb{N} \tag{3.3}$$
$$y_j = h \times H_{row},\ h \in \mathbb{N} \tag{3.4}$$
$$y_j = y_k \Rightarrow x_j + W_j \leq x_k \tag{3.5}$$

To avoid that significant disruptions may impair the quality of the solutions provided by upstream optimization steps (KIM; HU; VISWANATHAN, 2014), Equation (3.6) specifies an upper bound D_{max} for the displacement of every $c_j \in \mathcal{C}$ with respect to its location (x_j^0, y_j^0) in the initial placement P.

$$|x_j - x_j^0| + |y_j - y_j^0| \leq D_{max} \tag{3.6}$$

To avoid major changes that may compromise the quality of solutions in upcoming optimization steps (e.g. routing), upper bounds are defined for **signal wirelength** and **placement density** increase, as follows.

Let $w^0(n)$ and $w(n)$ denote, respectively, the wirelengths of a signal net $n \in \mathcal{N}$ in the initial placement P and in the new placement P^*. Equation (3.7) specifies an upper bound for signal wirelength increase.

$$\sum_{n \in \mathcal{N}} w(n) - \sum_{n \in \mathcal{N}} w^0(n) \leq SWI_{max} \tag{3.7}$$

To keep placement density under fine-grain control, the placement area is usually divided into regular bins whose sizes are all the same length: a multiple of the standard cell height, i.e. $k \times H_{row}$, with $k \in \mathbb{N}$. The density observed for every bin b_i, denoted as $d(b_i)$, is the ratio between the area occupied by the cells located in b_i and the total bin area (excluding the bins fully occupied by macro blocks). Usually, a target is specified for the maximum density expected for each bin, denoted as d_{target}, where a dense bin has $d(b_i) \geq d_{target}$. There are different metrics in the literature to assess the impact of dense bins on the overall placement density. To impose constraints on placement den-

sity, this work employs the well-known Average Bin Utilization (ABU) metric (KIM et al., 2012), as reviewed in the sequel. Let B_γ represent the set containing the γ% most dense bins in the placement area. For example, the set B_2 contains the densities of 2% most dense bins in the placement area. The placement density is then computed as follows. First, Equation (3.8) computes the ABU_γ as the average density of bins in the set B_γ. Then Equation (3.9) indicates the overflow for a given γ. Finally, Equation (3.10) computes the placement density as the weighted average of overflows for each $\gamma \in \Gamma$. Typical values employed for the set Γ are 2, 5, 10, and 20, while commonly used weights are $w_2 = 10,\ w_5 = 4,\ w_{10} = 2,\ w_{20} = 1$ (KIM; HU; VISWANATHAN, 2014).

$$ABU_\gamma = \frac{\sum_{b_i \in B_\gamma} d(b_i)}{|B_\gamma|} \tag{3.8}$$

$$overflow_\gamma = \max(0, \frac{ABU_\gamma}{d_{target}} - 1) \tag{3.9}$$

$$PD = \frac{\sum_{\gamma \in \Gamma}(w_\gamma \times overflow_\gamma)}{\sum_{\gamma \in \Gamma} w_\gamma},\ \Gamma = \{2,5,10,20\} \tag{3.10}$$

Let PD^0 and PD denote, respectively, the placement densities in the initial placement P and in the new placement P^*. Equation (3.11) specifies an upper bound for placement density increase.

$$PD - PD^0 \leq PDI_{max} \tag{3.11}$$

3.2.1.2 Choice of Objective Functions

The actual clock tree and its routing are built during the upcoming clock tree synthesis step. Therefore, to minimize the clock net wirelength, we have to rely on an estimate of the final clock tree routing, i.e. a **virtual clock tree**. A virtual clock tree is represented by a rooted tree $T(V,E)$. Except for the root node v_o, which represents the clock source, every $v \in V$, with $v \neq v_0$, represents either a sequential element or a Steiner point (KAHNG et al., 2011). Every edge $(p,c) \in E$ represents a clock net segment connecting a parent node $p \in V$ to its child $c \in V$. Let $w(p,v)$ denote the wirelength of the segment represented by (p,v). For **incremental register placement**, we employ the virtual clock tree wirelength as the objective function, according to

the equation below.

$$\sum_{(p,v)\in E} w(p,v) \tag{3.12}$$

In the following, we present how we model the timing constraints required to define our objective function for **incremental TDP**.

Recall that late and early arrival times of cells can be recursively defined from timing startpoints towards timing endpoints, as follows:

$$a_j^L = \max_{i\in\mathcal{I}_j}(a_i^L + d_{i,j}^L) \qquad a_j^E = \min_{i\in\mathcal{I}_j}(a_i^E + d_{i,j}^E) \tag{3.13}$$

Also recall that late and early slacks are tracked at circuit timing endpoints to measure the circuit timing violations, as below:

$$slk_j^L = r_j^L - a_j^L \qquad slk_j^E = a_j^E - r_j^E,\ \forall j \in \mathcal{TE} \tag{3.14}$$

Timing optimization techniques such as ITDP, typically try to improve a timing metric known as worst negative slack (WNS) which captures the most severe violation (as a negative value corresponding to the timing endpoint with worst slack) or non-violation (as a zero valued upper bound). Although the WNS metric represents timing violations on the worst path, a circuit may have several other near-critical paths violating the timing constraints.

That is why this work employs, as objective function for **ITDP**, the metrics late and early total negative slack, as redefined in Equations (3.15) and (3.16).

$$TNS^L = \sum_{j\in\mathcal{TE}} \min(0, slk_j^L) \tag{3.15}$$

$$TNS^E = \sum_{j\in\mathcal{TE}} \min(0, slk_j^E) \tag{3.16}$$

3.2.2 Casting ITDP into an LR Formulation

Lagrangian Relaxation (LR) is an effective technique to tackle problems with hard constraints. The idea is to solve a new problem, called LR problem, whose hard constraints are incorporated into the objective function, as penalty terms, weighted by coefficients (λ) known as Lagrange Multiplier (LM) (OZDAL; BURNS; HU, 2012). The LR

problem is addressed by iteratively solving two subproblems: 1) the Lagrangian Relaxation Subproblem (LRS) that targets the minimization of the new objective function, also known as Lagrangian function, for a set of fixed LMs. 2) the Lagrangian Dual Problem (LDP) that targets the update of LMS to maximize the solution from LRS. Figure 24 illustrates how the primal problem is cast into an LR formulation.

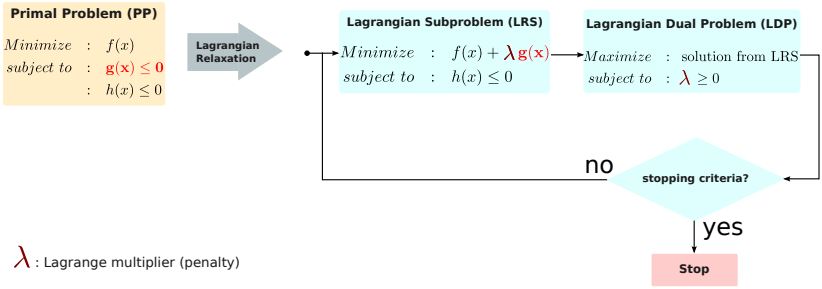


Figure 24: **Overview of Lagrangian Relaxation.** *Assume that $g(x) \leq 0$ represents a given hard constraint for the Primal Problem (PP). Therefore, these hard constraints are relaxed into the new objective function multiplied by the penalty term λ. The LRS and LDP are then solved iteratively until a given stopping criteria, e.g. the gap between the primal and dual is small enough or a maximum number of iterations is reached.*

First, to ensure that only non-positive slack values are accounted for in the objective function we introduce late **negative** slack, denoted $slk_j^{L'} = \min(0, slk_j^L)$, and early **negative** slack, denoted $slk_j^{E'} = \min(0, slk_j^E)$. Therefore, the TDP formulation can be rewritten as a minimization problem (Equation 3.17). We introduce the set of inequality constraints in Equations (3.18) to (3.20) in order to model the timing information in the circuit, as follows:

$$\textbf{Min.} \quad : \quad -\sum_{j \in \mathcal{TE}} slk_j^{L'} - \sum_{j \in \mathcal{TE}} slk_j^{E'} \tag{3.17}$$

$$\textbf{S.t.} \quad : \quad slk_j^{L'} \leq 0 \text{ and } slk_j^{E'} \leq 0, \ \forall j \in \mathcal{TE} \tag{3.18}$$

$$: \quad slk_j^{L'} \leq r_j^L - a_j^L \text{ and } s_j^{E'} \leq a_j^E - r_j^E, \ \forall j \in \mathcal{TE} \tag{3.19}$$

$$: \quad a_i^L + d_{i,j}^L \leq a_j^L \text{ and } a_i^E + d_{i,j}^E \geq a_j^E, \ \forall c_j \in \mathcal{C} \tag{3.20}$$

$$: \quad (3.1) \text{ to } (3.6) \tag{3.21}$$

Equation (3.18) ensures that $slk_j^{L'}$ and $slk_j^{E'}$ only assume neg-

ative values, as we do not want to account for positive slacks in the objective function. Equation (3.19) states that late/early slacks must be greater than or equal to late/early negative slacks. Finally, Equation (3.20) defines late and early arrival times for each cell. The objective function in Equation (3.17) is also subject to legality and maximum displacement constraints, as captured by Equations (3.1) to (3.6).

Our idea is to relax the late and early timing constraints and incorporate them into the objective function, similarly to what has been proposed for gate sizing (OZDAL; BURNS; HU, 2012). The inequalities modeling late and early negative slacks at timing endpoints (Equation 3.18) are accompanied by non-negative Lagrange multipliers $\lambda^{L'}$ and $\lambda^{E'}$, respectively. The remaining late and early timing constraints are accompanied by non-negative Lagrange multipliers λ^{L} and λ^{E}, respectively. Therefore, each LM represents a net-weight indicating the criticality of the net i, j from the output of c_i to the output c_j. This leads to the following Lagrangian function, which incorporates the relaxed timing constraints:

$$\begin{aligned} L_\lambda : & - \sum_{j \in \mathcal{TE}} slk_j^{L'} - \sum_{j \in \mathcal{TE}} slk_j^{E'} + \sum_{j \in \mathcal{TE}} \lambda_j^{L'} slk_j^{L'} + \sum_{j \in \mathcal{TE}} \lambda_j^{E'} slk_j^{E'} \\ & + \sum_{j \in \mathcal{TE}} \lambda_j^{L} (slk_j^{L'} - r_j^L + a_j^L) + \sum_{j \in \mathcal{TE}} \lambda_j^{E} (slk_j^{E'} - a_j^E + r_j^E) \\ & + \sum_{c_j \in \mathcal{C}} \Big(\sum_{i \in \mathcal{I}_j} \lambda_{i,j}^{L} (a_i^L + d_{i,j}^L - a_j^L) + \sum_{i \in \mathcal{I}_j} \lambda_{i,j}^{E} (a_j^E - a_i^E - d_{i,j}^E) \Big) \end{aligned} \quad (3.22)$$

The gate sizing technique from Ozdal, Burns and Hu (2012) exploited a few flow conservation conditions to eliminate the slack variables and thus simplify the Lagrangian function. We apply similar flow conservation to simplify L_λ. First, the late negative slack variables $slk_j^{L'}$ cancel out if $\forall j \in \mathcal{TE},\ \lambda_j^{L'} + \lambda_j^{L} = 1$. Early negative slack variables $slk_j^{E'}$ also cancel out if $\forall j \in \mathcal{TE},\ \lambda_j^{E'} + \lambda_j^{E} = 1$. From such simplification, we obtain the following Lagrangian function:

$$\begin{aligned} L_\lambda : & \sum_{j \in \mathcal{TE}} \lambda_j^{L} (-r_j^L + a_j^L) + \sum_{j \in \mathcal{TE}} \lambda_j^{E} (-a_j^E + r_j^E) \\ & + \sum_{c_j \in \mathcal{C}} \Big(\sum_{i \in \mathcal{I}_j} \lambda_{i,j}^{L} (a_i^L + d_{i,j}^L - a_j^L) + \sum_{i \in \mathcal{I}_j} \lambda_{i,j}^{E} (a_j^E - a_i^E - d_{i,j}^E) \Big) \end{aligned} \quad (3.23)$$

The minimization of the Lagrangian function above can yield $L_\lambda = -\infty$ if the Lagrange multipliers do no satisfy certain conditions (WANG; DAS; ZHOU, 2009; LEE; GUPTA, 2012). This can be readily seen by reorganizing Equation (3.23) into Equation (3.24).

$$\begin{aligned} L_\lambda : & - \sum_{j \in \mathcal{TE}} \lambda^L_j r^L_j + \sum_{j \in \mathcal{TE}} \lambda^E_j r^E_j \\ & + \sum_{c_j \in \mathcal{C}} \Big(\sum_{k \in \mathcal{O}_j} \lambda^L_{j,k} - \sum_{i \in \mathcal{I}_j} \lambda^L_{i,j} \Big) a^L_i + \sum_{c_j \in \mathcal{C}} \Big(\sum_{k \in \mathcal{O}_j} \lambda^E_{j,k} - \sum_{i \in \mathcal{I}_j} \lambda^E_{i,j} \Big) a^E_i \\ & + \sum_{c_j \in \mathcal{C}} \Big(\sum_{i \in \mathcal{I}_j} \lambda^L_{i,j} d^L_{i,j} \Big) + \sum_{c_j \in \mathcal{C}} \Big(\sum_{i \in \mathcal{I}_j} \lambda^E_{i,j} d^E_{i,j} \Big) \end{aligned} \tag{3.24}$$

Observe that, if the arrival time coefficients (i.e. the difference of LMs summations) are non-zero the problem is unbounded. To avoid such cases, it can be simplified by introducing flow conservation conditions to ensure that all arrival time coefficients are zero. Equations (3.25) and (3.26) formalize such flow conservation for late and early constraints, respectively. Figure 25 illustrates how the flow conservation conditions[2] are applied considering a given cell c_i.

$$\sum_{j \in \mathcal{O}_k} \lambda^L_{j,k} - \sum_{i \in \mathcal{I}_j} \lambda^L_{i,j} = 0 \tag{3.25}$$

$$\sum_{j \in \mathcal{O}_k} \lambda^E_{j,k} - \sum_{i \in \mathcal{I}_j} \lambda^E_{i,j} = 0 \tag{3.26}$$

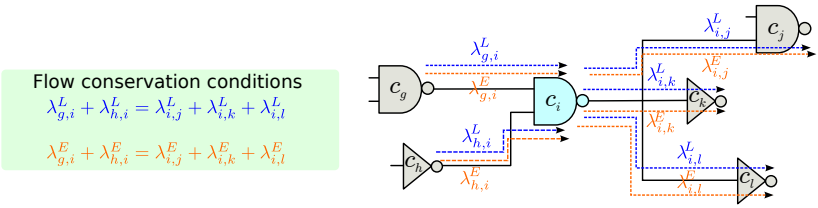


Figure 25: **Flow conservation conditions for net-weights.** *The flow conservation conditions for each cell imply that the sum of late and the sum of early Lagrange multipliers from input nets must be equal to the sums from output nets.*

[2]These flow conservation conditions are similar to Kirchhoff's first law, which states that the sum of the currents entering and leaving a node equals to zero.

Besides the flow conservation conditions, we also assume late and early required times as constants during a given LR iteration. This assumption is sound as far as a negligible amount of registers is moved by the proposed ITDP technique. Indeed, this is very likely because IRP is applied beforehand and the locations of non-critical registers are locked during ITDP. Therefore, the impact of the skew variation induced by a given ITDP iteration is likely to be marginal and can be ignored (within the scope of a given iteration only) for the sake of reducing ITDP runtime (the experimental confirmation of that hypothesis is presented in Figures 51 and 34). This leads to the final Lagrangian function to be minimized by the proposed technique:

$$L_\lambda : \sum_{c_j \in C} \left(\sum_{i \in \mathcal{I}_j} \lambda^L_{i,j}(d^L_{i,j}) + \sum_{i \in \mathcal{I}_j} \lambda^E_{i,j}(-d^E_{i,j}) \right) \tag{3.27}$$

The associated LRS minimizes the Lagrangian function L_λ by finding the location of each movable $c_j \in \mathcal{C}$, assuming a set of fixed net-weights (LMs), as presented in Equation (3.28). Then the LDP updates the net-weights (LMs) to maximize the solution from LRS (Q_λ), as shown in Equation (3.30). Therefore, the LRS and LDP problems are solved iteratively.

$$\textbf{LRS} \quad : \quad Q_\lambda = \min_{(x_j, y_j) \forall c_j \in \mathcal{C}} L_\lambda \tag{3.28}$$

$$: \quad (3.1) \text{ to } (3.6) \tag{3.29}$$

$$\textbf{LDP} \quad : \quad \max_{\lambda \geq 0} Q_\lambda \tag{3.30}$$

From the simplified Lagrangian function in Equation (3.27), we can conclude that, by minimizing the weighted summation of late (early) delay and late (early) net-weights, the metrics TNS^L and TNS^E are also minimized.

The LRS problem specified by Equations (3.28) and (3.29) is an integer programming problem that is solved using a discrete local search, as detailed in Section 3.3.4.1. The LDP problem specified by Equation (3.30) is a convex optimization problem (BOYD; VANDENBERGHE, 2004), but it is non-differentiable (WANG; DAS; ZHOU, 2009). Therefore, it can be solved using the subgradient algorithm, as described in Section 3.3.4.1.

3.3 PROPOSED TECHNIQUES

This section presents the proposed approach to solve the clock-tree-aware incremental timing-driven placement problem through a decomposition into subproblems. First, Section 3.3.1 details the mechanism and strategies employed to handle legalization during IRP and ITDP. Then Section 3.3.2 presents the proposed flow and the appropriate order to avoid that the solutions to the subproblems disrupt each other. Section 3.3.3 details the proposed technique for IRP. Finally, Section 3.3.4 presents the proposed techniques for ITDP.

3.3.1 Handling of Legalization During Incremental Placement

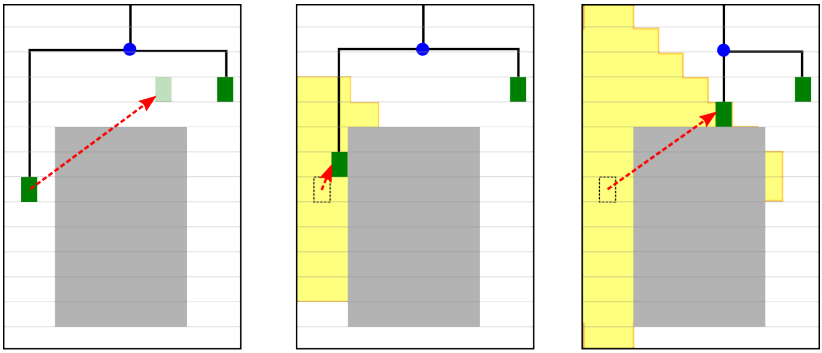


Figure 26: **Example of how the reshaped legal area (yellow) excludes macro blocks (gray) and maps a candidate location outside the reshaped legal area to a location inside it.** *(a) The original target location. (b) Placing the cell under a short displacement constraint. (c) Placing the cell under a long displacement constraint.*

Recall that during incremental placement techniques, to avoid significant disruptions that may impair the quality of the solutions provided by upstream optimization steps, there is an upper bound for the displacement of each cell with respect to its initial location, as defined in the constraint from Equation (3.6). The area in which the cell can be moved without violating the maximum displacement constraint is referred to as **legal area**.

Besides, the cell relocations induced by IRP and ITDP can cause overlaps between cells and also between cells and macro blocks. There-

fore, we maintain a reshaping mechanism active during the whole optimization process so as to exclude macro blocks from the cell's legal area. As a result, legal areas are permanently available for the circuit cells under relocation, as illustrated in Figure 26. That figure shows how a candidate location outside the **reshaped legal area** is mapped to a location inside it assuming two distinct scenarios, depending on how tight the displacement constraint is. The green rectangles represent registers and the blue point represents a parent node in the clock tree (Steiner point). Suppose a target location (shaded green rectangle) for the register (in the left), as shown in Figure 26 (a). In a short displacement scenario (Figure 26 (b)), since the macro block area is excluded from the reshaped legal area, the register ends up on the boundary of that area, avoiding an overlap with the macro block. In a long displacement scenario (Figure 26 (c)), the maximum displacement area is large enough for the register to be placed on the other side of the macro block, closer to the target location.

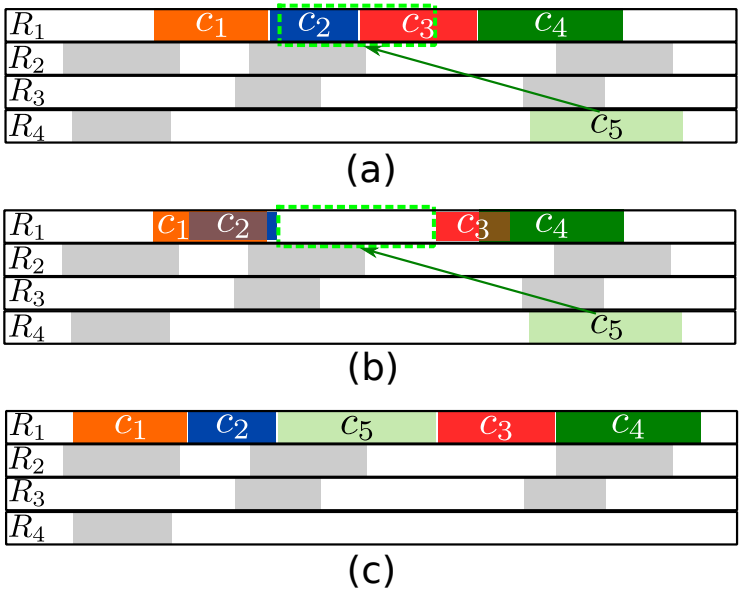


Figure 27: **Example of how the incremental legalization works.** *(a) Cell c_5, located in row R_4, targets a new location in R_1. (b) Since that new location overlaps with c_2 and c_3, those cells are shifted to the left and to the right-hand side, respectively. (c) Finally, c_1 and c_4 are also shifted and c_5 is placed in the target location. Obtained from (NETTO; LIVRAMENTO et al., 2016b)*

Although the reshaped legal area guarantees that cell locations do not overlap with macro blocks, there might occur overlaps between cells. Therefore, we apply the incremental legalization technique proposed in (NETTO; LIVRAMENTO et al., 2016b) after each cell movement, as exemplified in Figure 27. Appendix A describes the adopted incremental legalization technique. The advantage of applying incremental legalization is the possibility of revoking a would-be relocation as soon as it is clear that it breaks the maximum displacement constraint and cannot be made legal through mapping. Therefore, the combination of incremental legalization and mapping of candidate locations provides guarantees that the final result is legal and obeys displacement constraints.

3.3.2 The Proposed Flow

Since the proposed approach solves the clock-tree-aware incremental timing-driven placement problem via decomposition, the question becomes what subproblem should be solved first. As the percentage of registers in a circuit is significant (15% on average, according to (CHEON et al., 2005; LEE; MARKOV, 2012)), a large number of cells can be expected to be relocated during register placement to obtain a compact clock tree. This is likely to largely affect routability and circuit timing (as a result of the re-wiring between sequential and combinational elements). As a consequence, it would not be pragmatic to solve ITDP first because IRP would largely touch the standard cells (this would probably require ITDP to be applied anew). On the other hand, after IRP, only the registers in critical paths would be touched by ITDP. Therefore, when ITDP is applied after register placement, little impact can be expected on clock tree wirelength. Since the latter order reduces the influence between optimization subproblems, that is the ordering adopted in the proposed approach, whose overview is illustrated in Figure 28.

The incremental register placement subproblem is solved iteratively in five steps. First, all registers are unlocked and a virtual clock tree is created to guide the register relocations. Then the compact clock tree function (algorithm to be described in Section 3.3.3) relocates the registers. Since this step moves a large number of cells, it has a major impact on signal wirelength and placement density, which may affect the circuit routability. To keep registers out of congested areas and to try to recover from signal wirelength degradation, our technique locks

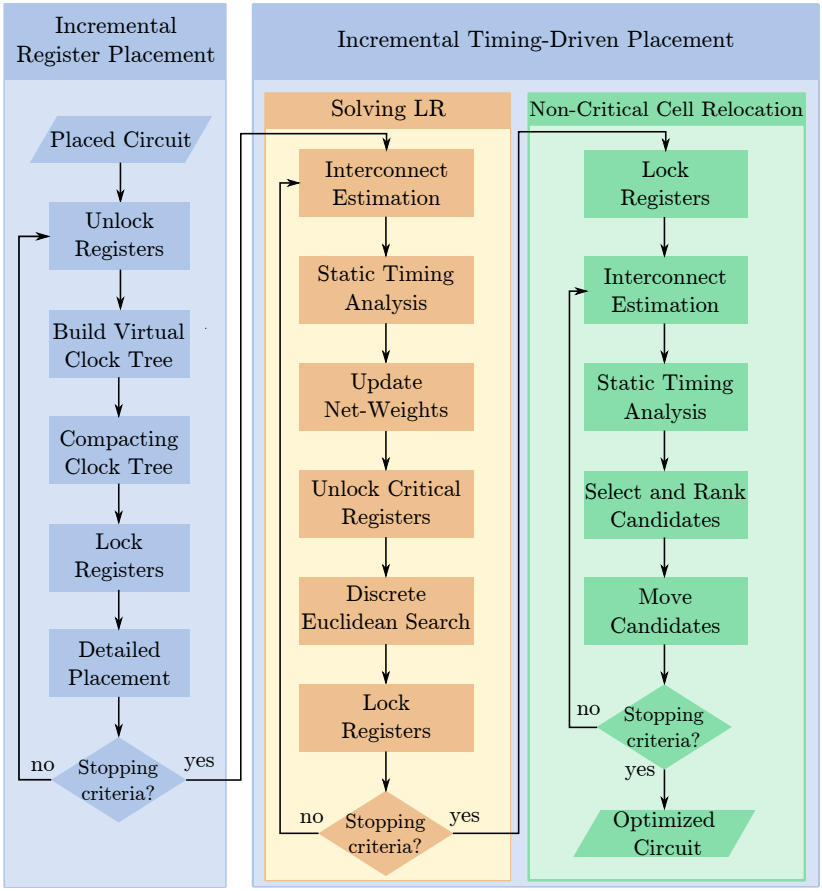


Figure 28: **Overview of the proposed approach.** *Since the proposed technique is incremental, the input comes from a global placement step, while the output can be used for clock tree synthesis. Interconnection estimates are used to evaluate wirelengths and density (no router was used to measure their actual values).*

registers before applying a detailed placement step (POPOVYCH et al., 2014; CHOW et al., 2014) to combinational cells only. This is performed iteratively until a stopping criteria is reached. We consider as stopping criteria a predefined maximum number of iterations (max_it) and upper bounds on the increase of signal wirelength (SWI_{max}) and placement density (PDI_{max}), according to Equations (3.7) and (3.11).

The work from Luo et al. (2008) highlighted the importance of moving sequential cells for time borrowing under imbalanced latches. In the proposed technique, we ensure that after register placement only critical registers are movable, allowing to relocate imbalanced latches from a positive slack side to a negative slack side. By doing so, we can improve the timing during ITDP with little register moves, keeping the quality of the previously obtained register placement solution. The proposed ITDP technique is divided in two subflows. Each subflow is iterated until the same stopping criteria is met, which is the same from IRP. **Solving LR** aims to solve the proposed LR formulation from Section 3.2.2 and has six steps. The first step estimates interconnections through the Steiner tree model proposed in (CHU; WONG, 2008). Then a static timing analysis tool updates the circuit timing information and the net-weights are updated to guide cell relocations. In the sequel, the critical registers are unlocked and cells are relocated using the Discrete Euclidean Search algorithm that will be detailed in Section 3.3.4.1. Finally, the registers are locked again until the next timing analysis is performed to identify the new critical registers. **Non-Critical Cell Relocation** aims to exploit the optimization potential left behind by the Solving LR technique. The basic idea is to use an accurate interconnection model to guide relocations of non-critical cells and reduce the capacitive load of critical cells, as detailed in Section 3.3.4.2.

For simplicity, the algorithms described in the next sections assume that max_it, SWI_{max}, and PDI_{max} all have global scope. That is why their declarations are omitted.

3.3.3 Incremental Register Placement

Before describing the algorithm underlying the proposed technique, it is interesting to mark the four main differences between the proposed approach and the most recent work on incremental register placement (LEE; MARKOV, 2012): 1) As opposed to that work, which employs a single, final detailed placement step, the proposed strat-

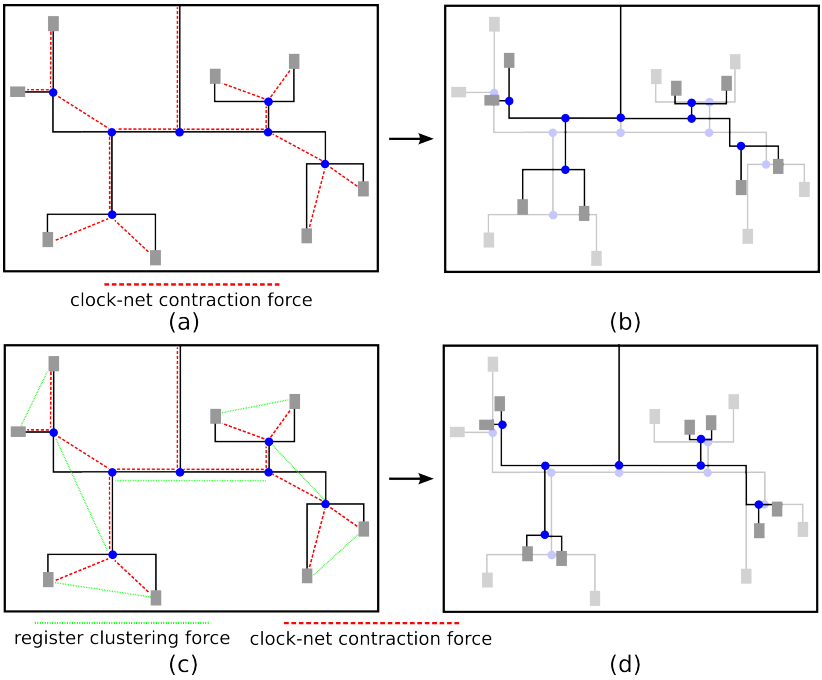


Figure 29: **Comparison between different contraction forces.** *(a) The use of clock-net contraction forces (red lines) from Lee and Markov (2012) and (b) the proposed strategy that combines them with register clustering forces (green lines). Although clock-net contraction forces tend to shift the clock tree towards the source, it barely reduces the capacitance at leaf level, as illustrated in (c). The proposed strategy not only contracts the clock tree, but also clusters registers to exploit path sharing for further wirelength reduction, as in (d).*

egy performs multiple detailed placement steps (with register locations locked) during incremental register placement so as to keep better control of density and signal wirelength. 2) As opposed to that work, which legalizes cells as a final step, the proposed technique performs on-the-fly legalization, thereby providing a more precise guidance towards clock tree compaction. 3) Unlike that work, which employs only forces between parent and child nodes of the clock tree (clock-net contraction forces), as illustrated in Figure 29 (a) and (b), the proposed strategy combines them with forces between sibling nodes (register clustering forces) (WANG et al., 2007), as illustrated in Figure 29 (c) and (d). The motivation to also exploit clustering forces lies in the fact that most

of the clock-tree capacitance (about 80%) is at leaf level (CHEON et al., 2005; GUTHAUS; WILKE; REIS, 2013). Besides, closer siblings improve register clustering, which favors further wirelength reductions due to path sharing. 4) As opposed to that work, which employs a deferred merge embedding algorithm to build a zero-skew tree, the proposed technique generates a virtual clock tree in the form of Steiner trees. It is worth mentioning that the adoption of a Steiner-based clock tree was required for compliance with the experimental infrastructure (see Sections B.1 and 3.4.1). Nevertheless, the proposed method provides support for using any clock routing algorithm.

Algorithm 1: INCREMENTAL_REGISTER_PLACEMENT

```
   Input  : P and α
   Output: P'
1  it ← 1;
2  while SWI ≤ SWI_max and PDI ≤ PDI_max and it ≤ max_it do
3      unlock registers;
4      tree ← build virtual clock tree;
5      COMPACTING_CLOCK_TREE(tree, reshaped legal areas, α);
6      lock registers;
7      apply detailed placement;
8      it ← it + 1; SWI ← signal wirelength increase; PDI ← placement
         density increase;
9  end
10 Function COMPACTING_CLOCK_TREE(tree, α)
11     foreach node in tree do
12         target ← (parent(node).location + Σ_{s∈siblings(node)} s.location) / (1 + |siblings(node)|);
13         movement_vector ← target − node.location;
14         new_location ← node.location + movement_vector × α;
15         if new_location is outside of reshaped legal area then
16             new_location ← intersection of movement_vector with the
                 boundaries of reshaped legal area;
17         end
18         move node to new_location;
19         if node is leaf then
20             place node in new_location and legalize;
21         end
22     end
```

Algorithm 1 describes the proposed incremental register placement technique. Given a relocation step α, it produces a new placement P' from the original global placement P. The top loop (lines 2-9) performs the clock tree compaction until a stopping criteria is reached. Initially, registers are unlocked (line 3) to allow for register relocations. Then after building a virtual clock tree (line 4), a compaction function is invoked (line 5). Then registers are locked (line 6) so as to apply detailed placement on combination elements only (line 7). A detailed placement technique combining features of two state-of-the-art works was devised (POPOVYCH et al., 2014; CHOW et al., 2014). Notice

that, as opposed to Lee and Markov (2012), where detailed placement is applied as a single step after the whole register placement is done, the proposed technique does it iteratively to keep a more precise tracking of signal wirelength and placement density (line 7).

The compaction function (lines 10-22) relies on a force-directed placement method to iteratively move each node to its ideal minimum-energy location (KAHNG et al., 2011). Such target location is obtained as the center of mass for parent and siblings (line 12). To allow for fine-grain control on legal relocations, the node is incrementally moved towards the target with a given step α (lines 13-18). If a candidate new location is outside the reshaped legal area, the node's final position is set to the closest location on its boundary (lines 15-18). Finally, when the node is a leaf, its relocation is committed and immediately legalized (lines 19-21).

The runtime complexity of the proposed IRP technique is dominated by two functions: compacting clock tree (line 5) and detailed placement (line 7). Since both functions invoke incremental legalization after each cell relocation, let us first analyze the factors affecting runtime complexity of the incremental legalization algorithm. The runtime complexity of each incremental legalization can be defined as $\mathcal{O}(n \log n)$, where n corresponds to the maximum number of cells in the row (as detailed in Appendix A). Since the maximum number of cells in a given circuit row is bounded by the cardinality of the set of cells $|\mathcal{C}|$, we can establish an asymptotic upper bound for each incremental legalization as $\mathcal{O}(|\mathcal{C}| \log |\mathcal{C}|)$. The **compacting clock tree** function visits each node in the clock tree, computes the average between parent and sibling locations, and performs a relocation. Despite the fact that a Rectilinear Steiner Minimum Tree contains at most $p-2$ Steiner points to connect p pins (KAHNG et al., 2011), only the leaf nodes are legalized (i.e. the sequential elements). Therefore, the asymptotic upper bound can be defined as $\mathcal{O}(|\mathcal{S}| \cdot |\mathcal{C}| \log |\mathcal{C}|)$, where $|\mathcal{S}|$ corresponds to the cardinality of the set of sequential cells. The **detailed placement** step visits each combinational cell, samples a constant number κ of candidate locations in a given neighborhood, invokes incremental legalization, and selects that location with best cost (in terms of signal wirelength and density). As a result, the runtime complexity can be stated as $\mathcal{O}(\kappa \cdot |\mathcal{NS}| \cdot |\mathcal{C}| \log |\mathcal{C}|)$, where $|\mathcal{NS}|$ corresponds to the cardinality of the set of non-sequential cells. Since the number of candidate locations κ is constant, the asymptotic worst-case complexity of the detailed placement step is $\mathcal{O}(|\mathcal{NS}| \cdot |\mathcal{C}| \log |\mathcal{C}|)$.

Therefore we can conclude that the worst-case runtime is bounded

by $\mathcal{O}(|\mathcal{S}|\cdot|\mathcal{C}|\log|\mathcal{C}|+|\mathcal{NS}|\cdot|\mathcal{C}|\log|\mathcal{C}|)$. Since $|\mathcal{S}|+|\mathcal{NS}|=|\mathcal{C}|$, the asymptotic complexity is $\mathcal{O}(|\mathcal{C}|^2\cdot\log|\mathcal{C}|)$. Although the time complexity of each incremental legalization is linearithmic, in practice the observed runtime of each incremental legalization is roughly linear. Besides, as the number of cells in a circuit is generally well-distributed among the circuit rows, the actual legalization runtime is far below the theoretical one.

3.3.4 Incremental Timing-Driven Placement

The proposed technique ensures that, after IRP, only critical registers are movable, i.e. those with negative slacks. This allows for relocating imbalanced latches from a positive slack side to a negative slack side. By doing so, it can improve the timing during ITDP with few register moves, thereby preserving the quality of the previously obtained register placement solution. To solve ITDP, we propose two complementary techniques. The first technique (Section 3.3.4.1) aims to solve the LR formulation obtained in Section 3.2.2. and mainly targets cells belonging to the circuit critical paths. Then the second technique (Section 3.3.4.2) takes advantage of the optimization potential left behind by the first one by exploiting a more accurate interconnection model to guide the relocations of non-critical cells. Algorithm 2 describes the high-level functions used during ITDP. Given a placement P', it produces an optimized placement P^* with less violations for both timing constraint scenarios (late and early).

Algorithm 2: INCREMENTAL_TIMING-DRIVEN_PLACEMENT

```
   Input  : P'
   Output: P*
1  SOLVE_LR() ;                              // invoke Algorithm 3
2  NON-CRITICAL_CELL_RELOCATION() ;          // invoke Algorithm 5
```

3.3.4.1 Solving the Proposed LR Formulation

Algorithm 3 presents the proposed technique to solve the LR formulation. Initially, net-weight vectors are initialized for both scenarios (line 1). The algorithm's main loop has 6 major steps that perform timing-aware relocations until a stopping criteria is reached (lines 2-9). First, Steiner Trees are employed as estimates for the circuit's intercon-

nection wirelengths (line 3). This is followed by static timing analysis step (line 4) to update circuit timing information. Then net-weight vectors are updated to reflect the new timing information (line 5) and critical registers are unlocked based on the timing analysis report (line 6). Finally, a Discrete Euclidean Search is performed to find new cell locations (line 7) relying on the updated net-weights. The novel Discrete Euclidean Search is explained later on (see Algorithm 4), whereas the vector-update mechanism (lines 10-29), which is an adaptation of the sub-gradient method proposed in (TENNAKOON; SECHEN, 2008), is described in the following. The key idea is to iteratively scale net-weights up and down proportionally to the severity of timing violations. First, net-weight scaling is performed for every timing endpoint (lines 11-14) and then for each input/output pair of every cell (lines 15-20). Finally, to satisfy flow conservation conditions (see Section 3.2.2), net-weights are redefined such that the sum of the output weights of each cell is proportionally distributed to each input (lines 21-28). This is performed by visiting cells in reverse topological order.

Let us stress the main advantages of using Lagrangian multipliers as net-weights.

1. **Capture of temporal criticality for every net**: If a net has been historically critical, the value of its multiplier is kept high even if a slack variation is observed during some iteration. This prevents a net from oscillating from critical to non-critical, which would impair efficiency.

2. **Capture of spatial criticality for every net**: Due to the flow conservation conditions, the value of a multiplier correlates with the number of critical paths passing through a net (KAHNG; WANG, 2004). Since a slack represents the worst path passing through a cell (REN; PAN; KUNG, 2005), the effectiveness of a technique based only on slacks would be limited, because the effect of path sharing would be overlooked when guiding relocations.

Instead of updating timing information after each cell movement (and cope with prohibitively long runtimes for large circuits), Algorithm 3 performs timing analysis only at the beginning of each iteration. Although the timing inaccuracy between two successive updates may lead to bad-quality cell movements, two complementary features of the proposed technique alleviate the negative impact of the adopted update policy: 1) Steiner trees are used to obtain accurate estimates of interconnection capacitance and resistance (instead of simpler schemes

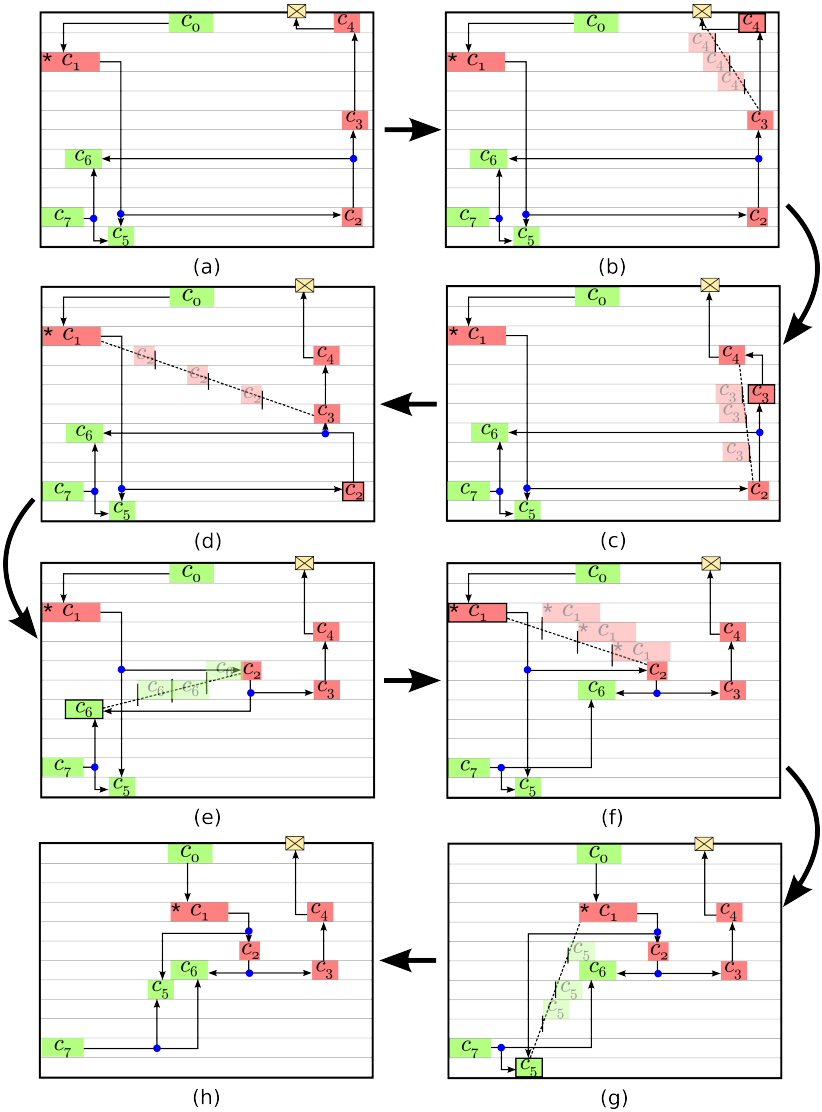


Figure 30: **How the Discrete Euclidean Search works.** *The circuit is composed of an output pad at the top right (yellow), 4 critical cells (red) and 4 non-critical ones (green). Among them, there is a single register, which is marked with an asterisk. Blue dots on interconnection denote Steiner points.*

Algorithm 3: SOLVE_LR

$it \leftarrow 1$; Initialize net-weights vectors $\vec{\lambda^L}$ and $\vec{\lambda^E}$;
while $SWI \leq SWI_{max}$ *and* $PDI \leq PDI_{max}$ *and* $it \leq max\_it$ **do**
  Interconnect estimation;
  Static Timing Analysis to update circuit timing information;
  $(\vec{\lambda^L}, \vec{\lambda^E}) \leftarrow$ `UPDATE_NET_WEIGHTS`$(\vec{\lambda^L}, \vec{\lambda^E})$;
  unlock critical registers;
  `DISCRETE_EUCLIDEAN_SEARCH`$(\vec{\lambda^L}, \vec{\lambda^E})$; `// invoke Algorithm 4`
  lock registers;
end
Function `UPDATE_NET_WEIGHTS`$(\vec{\lambda^L}, \vec{\lambda^E})$
  foreach $j \in \mathcal{TE}$ **do**
    $\lambda_j^E \leftarrow \lambda_j^E \times \frac{r_j^E}{a_j^E}$;
    $\lambda_j^L \leftarrow \lambda_j^L \times \frac{a_j^L}{r_j^L}$;
  end
  foreach $c_j \in \mathcal{C}$ **do**
    foreach $i \in \mathcal{I}_j$ **do**
      $\lambda_{i,j}^E \leftarrow \lambda_{i,j}^E \times \frac{a_j^E}{a_i^E + d_{i,j}^E}$;
      $\lambda_{i,j}^L \leftarrow \lambda_{i,j}^L \times \frac{a_i^L + d_{i,j}^L}{a_j^L}$;
    end
  end
  foreach $c_j \in \mathcal{C}$ *in reverse topological order* **do**
    $\mu_j^E = \sum_{i \in \mathcal{I}_j} \lambda_{i,j}^E$;
    $\mu_j^L = \sum_{i \in \mathcal{I}_j} \lambda_{i,j}^L$;
    foreach $i \in \mathcal{I}_j$ **do**
      $\lambda_{i,j}^E \leftarrow \frac{\lambda_{i,j}^E}{\mu_j^E} \times \sum_{k \in \mathcal{O}_j} \lambda_{j,k}^E$;
      $\lambda_{i,j}^L \leftarrow \frac{\lambda_{i,j}^L}{\mu_j^L} \times \sum_{k \in \mathcal{O}_j} \lambda_{j,k}^L$;
    end
  end
  return $(\vec{\lambda^L}, \vec{\lambda^E})$;

like HPWL); therefore, gate delays are accurately updated after each cell movement; 2) since Lagrange multipliers capture temporal criticality for every net, it can be expected that the high-value multipliers of historically-critical nets should limit the number of bad-quality cell movements until the beginning of the next iteration.

Before describing the proposed algorithm for the novel Discrete Euclidean Search, let us illustrate the key idea by means of an example. Figure 30 illustrates how the proposed discrete search selects which elements should be relocated (candidate cells) and where they

could be moved in (candidate locations). In the example, the connection topology is indicated through directed arrows to clearly distinguish the fanout and the fanin of each cell. A cell is selected as a candidate if it is a critical (red) cell or if it is non-critical (green) but it belongs to the fanout of a critical cell. Once the candidate cells are defined, they are visited in reverse topological order starting at the most critical timing endpoint (c4, c3, c2, c6, c1, c5). Each subfigure highlights a distinct cell being visited in that order. The search technique generates a discrete neighborhood for each visited cell. If the fanin and the fanout of that cell are both critical, as in Figure 30 (b), (c) and (d), then the technique samples a few (shaded) candidate locations along the Euclidean distance between fanin and fanout pin locations. If the visited cell has a critical cell either as fanin or fanout, the Euclidean distance is computed between the visited cell itself and that critical cell, as in Figure 30 (e), (f) and (g). Among the candidate locations for each visited cell, the technique selects the one that minimizes the Lagrangian function. Therefore, the selected relocations end up shortening and unloading the critical path, as depicted in Figure 30 (h).

Algorithm 4 describes the Discrete Euclidean Search. Initially, a set of candidate cells is defined (line 1). For each candidate cell, a set of candidate locations is obtained (line 4). Among those locations, the one with lowest cost is determined (lines 5-26). Finally, each candidate is placed in the selected location (line 26) and legalized. To accurately compute the impact of each candidate location on interconnection delay and capacitance, Steiner trees (CHU; WONG, 2008) are generated to calculate the output capacitances for each candidate cell (line 7) and every fanin (line 9). Three different contributions to the candidate location cost are computed: 1) the contribution of each fanin (lines 10-13), which is calculated as the product of the respective arc delay and net-weight (this reflects the impact of its new output capacitance C_i^{down}); 2) the contribution of the interconnection delay between each fanin and the respective candidate cell plus the arc delay of that cell (lines 14-15); and 3) the contribution of the interconnection delay between the candidate cell and each fanout (lines 18-19). Finally, the costs associated with late and early scenarios are cast into a single number to reflect the Lagrangian function (line 21). For the **selection of candidate cells** (lines 28-42), a predefined constant ρ specifies how many timing endpoints with negative slack are taken into account (line 29). Starting at each selected critical timing endpoint, the circuit is traversed in reverse topological order until a timing startpoint (line 33). A cell is selected as a candidate if it is critical (line 35) or if it is

Algorithm 4: DISCRETE_EUCLIDEAN_SEARCH

Input : $\vec{\lambda^L}$, $\vec{\lambda^E}$
Output: Cell relocations

1 $candidate\_cells \leftarrow$ SELECT_CANDIDATE_CELLS(ρ);
2 **foreach** $c_j \in candidate\_cells$ **do**
3 $best\_cost \leftarrow \infty$;
4 $discrete\_candidate\_locations \leftarrow$ SELECT_DISCRETE_CANDIDATE_LOCATIONS(c_j, $num\_samples$);
5 **foreach** $location \in discrete\_candidate\_locations$ **do**
6 $(cost^L, cost^E) \leftarrow (0, 0)$;
7 generate Steiner tree connected to the output of c_j and compute the output capacitance C_j^{down};
8 **foreach** $i \in \mathcal{I}_j$ **do**
9 generate Steiner tree connected to the output of c_i and compute the output capacitance C_i^{down};
10 **foreach** $h \in \mathcal{I}_i$ **do**
11 $cost^L \leftarrow cost^L + \delta^L_{h,i} \times \lambda^L_{h,i}$;
12 $cost^E \leftarrow cost^E + \delta^E_{h,i} \times \lambda^E_{h,i}$;
13 **end**
14 $cost^L \leftarrow cost^L + (\tau_{i,j} + \delta^L_{i,j}) \times \lambda^L_{i,j}$;
15 $cost^E \leftarrow cost^E + (\tau_{i,j} + \delta^E_{i,j}) \times \lambda^E_{i,j}$;
16 **end**
17 **foreach** $k \in \mathcal{O}_i$ **do**
18 $cost^L \leftarrow cost^L + \tau_{j,k} \times \lambda^L_{j,k}$;
19 $cost^E \leftarrow cost^E + \tau_{j,k} \times \lambda^E_{j,k}$;
20 **end**
21 $cost\_location \leftarrow cost^L + cost^E$;
22 **if** $(cost\_location < best\_cost)$ **then**
23 $(best\_location, best\_cost) \leftarrow (location, cost\_location)$
24 **end**
25 **end**
26 place c_j in the coordinate of $best\_location$ and legalize;
27 **end**
28 **Function** SELECT_CANDIDATE_CELLS(ρ)
29 $CTE \leftarrow \rho$ timing endpoints with worst negative slacks;
30 $candidate\_cells \leftarrow \emptyset$;
31 **foreach** $cte \in CTE$ **do**
32 $pin \leftarrow cte$;
33 **while** $pin \neq timing\_startpoint$ **do**
34 $c_j \leftarrow$ driver of interconnection connected to pin;
35 $candidate\_cells.push\_back(c_j)$;
36 **foreach** $k \in \mathcal{O}_j$ *such that* $s^L_k > 0$ **do**
37 $candidate\_cells.push\_back(c_k)$;
38 **end**
39 $pin \leftarrow$ input pin of c_j with worst slack;
40 **end**
41 **end**
42 **return** $candidate\_cells$;
43 **Function** SELECT_DISCRETE_CANDIDATE_LOCATIONS(c_j, $num\_samples$)
44 $pin_i \leftarrow$ output pin of most critical fanin $i \in \mathcal{I}_j$;
45 **if** $(s^L_i > 0)$ **then** $pin_i \leftarrow$ output pin of c_j;
46 $pin_k \leftarrow$ input pin of most critical fanout $k \in \mathcal{O}_j$;
47 **if** $(s^L_k > 0)$ **then** $pin_k \leftarrow$ input pin of c_j;
48 $discrete\_candidate\_locations \leftarrow$ sample $num\_samples$ over Euclidean distance between pin_i and pin_k locations;
49 **return** $discrete\_candidate\_location$;

non-critical but it belongs to the fanout of a critical cell (line 37). For the **generation of candidate locations** (lines 43-49), if the fanin and the fanout of that cell are both critical (the conditions in lines 45 and 47 are not satisfied), then the technique generates a predefined number of candidate locations along the Euclidean distance between fanin and fanout pin locations (line 48). If the visited cell has a non-critical cell either as fanin (line 44) or fanout (line 46), the Euclidean distance is computed between the visited cell itself and that critical cell.

The runtime of the proposed Discrete Euclidean Search is mainly affected by both the incremental legalization after each relocation (line 26) and the generation of Steiner trees (lines 7 and 9). Since in the worst-case scenario all the circuit cells are selected as candidates, the number of performed **incremental legalizations** is bounded by $\mathcal{O}(|\mathcal{C}|)$. Therefore, the incremental legalization alone leads to a runtime complexity of $\mathcal{O}(|\mathcal{C}|^2 \log |\mathcal{C}|)$. The number of **generated Steiner trees** during each Discrete Euclidean Search is $\mathcal{O}(|\mathcal{C}| \cdot num\_samples \cdot max(|\mathcal{I}|))$, where $max(|\mathcal{I}|)$ corresponds to the fanin set with maximum cardinality over all cells. Given that the maximum cardinality fanin set is constant, as it is bounded by the cell with most number of input pins in the library (and is rarely more than 4) (BHASKER; CHADHA, 2009), and the $num\_samples$ is also constant, thus the number of generated Steiner trees is also bounded by $\mathcal{O}(|\mathcal{C}|)$. Assuming that the runtime complexity of the Rectilinear Steiner Minimal Tree algorithm (CHU; WONG, 2008) is $\mathcal{O}(n \log n)$, where n corresponds to the net degree (i.e. number of points to be connected), thus the generation of Steiner trees corresponds to a complexity $\mathcal{O}(|\mathcal{C}| \cdot n \log n)$. As a consequence, we can establish an asymptotic worst-case complexity for the Discrete Euclidean Search as $\mathcal{O}(|\mathcal{C}|^2 \log |\mathcal{C}| + |\mathcal{C}| \cdot n \log n)$.

Although the time complexity of each incremental legalization is linearithmic, in practice the observed runtime of each incremental legalization is roughly linear. Besides, as the number of cells in a circuit is generally well-distributed among the circuit rows, the actual legalization runtime is far below the theoretical one. Another practical issue is that the number of candidate cells is far less than the total number of cells, leading to a runtime growth below the quadratic time complexity.

Albeit the proposed Discrete Euclidean Search makes use of an accurate model (i.e. Steiner tree model) to estimate interconnections resistances and capacitances, the candidate locations are sampled along the Euclidean distance between pin locations. Despite the effectiveness of the candidate locations along the Euclidean distance to shorten interconnections between critical cells and also between critical and

non-critical ones (as illustrated in Figure 30), they might not fully explore the opportunity to shorten non-critical interconnections. Figure 31 illustrates the room for optimization left by the Discrete Euclidean search when shortening a non-critical interconnection branch. Therefore, the next section presents the proposed technique that uses Steiner points to guide the relocation of non-critical cells to further reduce timing violations.

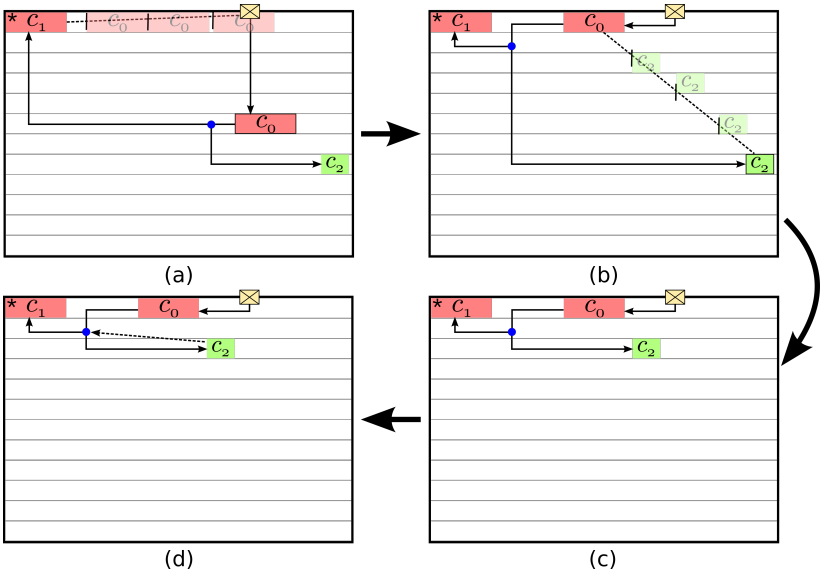


Figure 31: **Limitation of the Discrete Euclidean Search.** *The circuit is composed of an input pad at the top right (yellow), 2 critical cells (red) and 1 non-critical cell (green). Among them, there is a single register, which is marked with an asterisk. The blue dot denotes a Steiner point. (a) Relocation of critical cell* c_0 *and (b) non-critical cell* c_2*, and resulting solution (c). Observe in (d) that the Steiner point can be used as a target to further shorten the non-critical interconnection branch, thus reducing the capacitance of critical cell* c_0*.*

3.3.4.2 Non-Critical Cell Relocation

This section proposes a non-critical cell relocation technique that employs Steiner points as accurate candidate locations to reduce the capacitance of critical cells. As described in (LIVRAMENTO et al., 2015), the ratio of non-critical capacitance among critical interconnec-

tions may correspond to a large fraction, achieving more than 90% in some cases. Therefore, overlooking those non-critical branches would give up significant timing improvement oportunities.

Before describing the proposed Non-Critical Cell Relocation algorithm, let us illustrate the key idea by means of an example. Figure 32 illustrates how the proposed technique selects the candidate cells for relocation and how to find their respective candidate locations. In the example from Figure 32 (a), the connection topology is indicated through directed arrows to clearly distinguish the fanout of each cell. A cell is selected as a candidate if it is a non-critical (green) but it belongs to the fanout of a critical cell. Once the candidate cells are defined, the ranking of each candidate movement is performed. For instance, in Figure 32 (b), c_1 targets the Steiner point that lies over the macro block. Therefore, the reshaped legal are maps the candidate location to the border of the macro. That candidate location reduces the capacitance of the critical cell by 3.2fF and the delay by 0.64ps. The resulting movement is pushed into a priority-queue. Then Figures 32 (c) and (d) show how to compute the candidate movements for two of the remaining non-critical cells. Finally, Figures 32 (e) and (f) illustrate how the movements are performed (from the highest to the lowest delay reduction) and immediately legalized.

The proposed Non-Critical Cell Relocation technique is described in Algorithm 5. The top level functions (lines 2-5) are repeated until a stopping criteria is reached As a first step, a static timing analyzer is invoked (line 2) to update timing information. It relies on circuit routing estimates (Steiner trees). Then the candidate cells for shortening non-critical interconnection branches are selected (line 3) and ranked according to their potential to reduce the critical path delay (line 4). Finally, the cells are relocated to their target locations (line 5).

The **selection of candidate cells** (lines 7-21) is very similar to the one employed in the Discrete Euclidean Search. A predefined constant ρ specifies how many timing endpoints with negative slack are taken into account (line 8). Starting at each selected critical timing endpoint, the circuit is traversed in reverse topological order until a timing startpoint (line 12). A cell is selected as a candidate if it is non-critical fanout of a critical cell (lines 15-17). Finally, this function returns a set containing pairs of candidate cells and nets. The **ranking of candidate cells** (lines 22-35) aims to compute the potential of each candidate movement in reducing the respective critical cell delay and to rank them according to that potential. For each visited pair $(cell, net)$ (line 24), each Steiner point in the current net (line 26) is

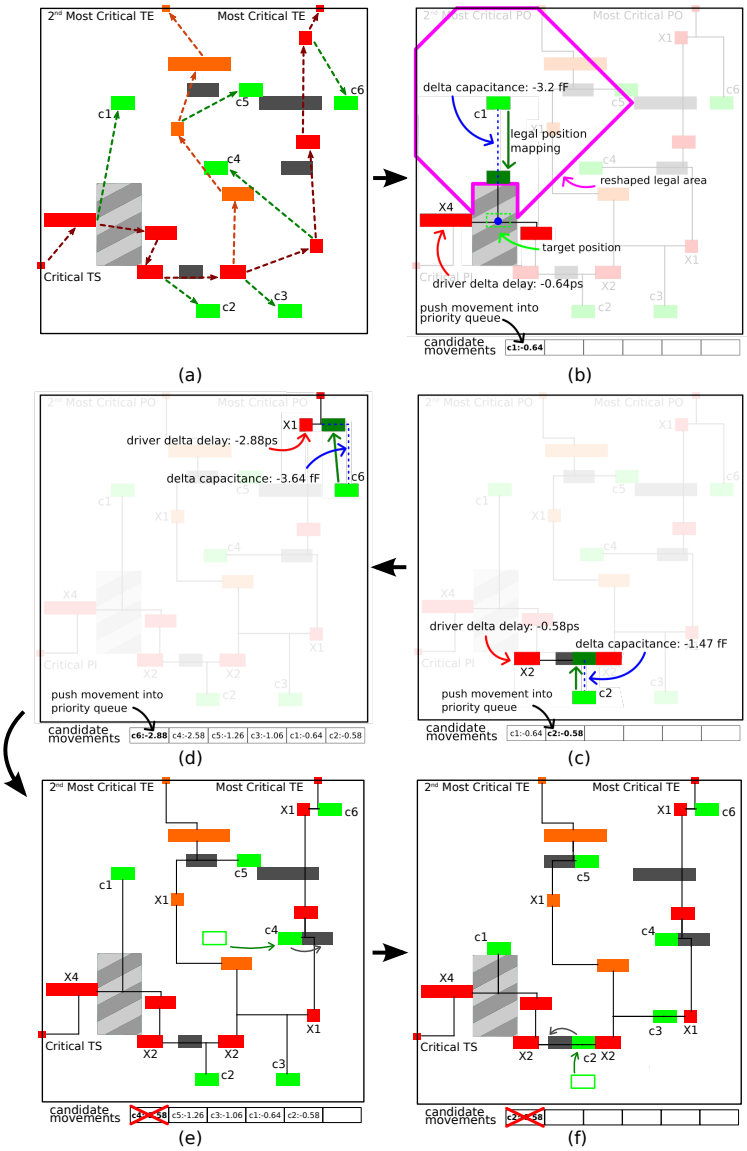


Figure 32: **How the Non-Critical Cell Relocation works.** *The circuit is composed of two critical paths, highlight in red and orange. The non-critical fanouts of critical cells (i.e. the candidates for relocation) are highlighted in green and marked from c_1 to c_6. (a) corresponds to the selection of candidate cells. (b), (c), and (d) show the ranking step while (e) and (f) illustrate the cell movements.*

Algorithm 5: NON-CRITICAL_CELL_RELOCATION

```
1  while SWI ≤ SWI_max and PDI ≤ PDI_max and it ≤ max_it do
2      Static Timing Analysis to update circuit timing information;
3      candidates_cells ← SELECT_CANDIDATE_CELLS();
4      movements ← RANK_CANDIDATE_CELLS(candidate_cells);
5      RELOCATE_CANDIDATE_CELLS(movements);
6  end
7  Function SELECT_CANDIDATE_CELLS()
8      CTE ← ρ timing endpoints with worst negative slacks;
9      candidate_cells ← ∅;
10     foreach cte ∈ CTE do
11         pin ← cte;
12         while pin ≠ timing_startpoint do
13             net ← interconnection connected to pin;
14             c_j ← driver of net;
15             foreach k ∈ O_j such that s_k^L > 0 do
16                 candidate_cells.push_back(c_k, net);
17             end
18             pin ← input pin of c_j with worst slack;
19         end
20     end
21     return candidate_cells;
22 Function RANK_CANDIDATE_CELLS(candidate_cells)
23     movements ← an empty priority-queue;
24     foreach (cell, net) ∈ candidate_cells do
25         curr ← current location of cell;
26         foreach target_location ∈ steiner_point(net) do
27             if target_location is outside of legal area then
28                 m ← line segment from curr to target_location;
29                 target_location ← intersection of m with the boundaries
                     of legal area;
30             end
31             potential ← potential of moving cell to target_location;
32             insert (potential, target_location, cell) in movements;
33         end
34     end
35     return movements;
36 Function RELOCATE_CANDIDATE_CELLS(movements)
37     while movements is not empty do
38         (potential, target, cell) ← extract best movement from
             movements;
39         place cell in the coordinate of target_location and legalize;
40     end
```

selected as a candidate location. If the target location lies outside the cell legal area, it is mapped to the respective intersection with the legal area boundaries (lines 28-29), as already exemplified in Figure 32 (b). Then line 31 computes the potential of reducing the net driver's delay when moving the cell to the target location. Finally, line 32 inserts the tuple $(potential, target\_location, cell)$ into the priority-queue *movements*, wherein the priority of a movement corresponds to its potential. The **relocation of candidate cells** visits the movements priority-queue, places each cell in the target location, and immediately legalizes (line 39) using the incremental legalization strategy from Sec-

tion 3.3.1. The advantage of legalizing each cell after a movement is that it is possible to evaluate the consequences immediately, allowing to undo it whenever the legalization is not possible. This strategy helps making the optimization process more predictable when compared to a single legalization step at the end of the optimization.

The runtime complexity of the proposed Non-Critical Cell Relocation is mainly affected by the ranking and relocation steps. The **ranking step** visits each net Steiner point (line 26) and computes its potential by generating a new Steiner tree to accurately compute the impact of moving the cell to the target location (line 31). The maximum number of Steiner points to connect a net with p pins is $p-2$ (KAHNG et al., 2011) and the runtime complexity to generate a Steiner Tree is $\mathcal{O}(n \log n)$, where n corresponds to the net degree (CHU; WONG, 2008). Therefore, the maximum number Steiner trees generated per candidate cell is $\mathcal{O}(p \cdot n \log n)$. Given that in the worst-case scenario all cells are selected as candidates, the asymptotic complexity of the ranking step is given by $\mathcal{O}(|\mathcal{C}| \cdot p \cdot n \log n)$. The **relocation step** performs an incremental legalization for each movement and is asymptotically bounded by $\mathcal{O}(|\mathcal{C}|^2 \log |\mathcal{C}|)$. Therefore, the algorithm has a worst-case complexity $\mathcal{O}(|\mathcal{C}|^2 \log |\mathcal{C}| + |\mathcal{C}| \cdot p \cdot n \log n)$

Although the time complexity is dominated by the incremental legalization, we observed in practice that the ranking step consumes most of the runtime, as the number of candidate cells is far less than $|\mathcal{C}|$.

3.4 EXPERIMENTAL VALIDATION

This section presents the experimental validation of the proposed techniques under the ICCAD 2015 ITDP Contest Infrastructure. (As a complement, Appendix B reports the experiments under the ICCAD 2014 ITDP Contest Infrastructure.) The algorithms were implemented in C++ and the experiments were performed on a Linux workstation with two CPUs Intel®Xeon®E5-5620 running at 2.4 GHz with 12GB RAM.

Section 3.4.1 details the ICCAD 2015 Contest experimental infrastructure, which overcomes the well-known clock-tree routing limitation affecting the 2014's version. Then Section 3.4.2 compares the results of the proposed approach with the best results available from the techniques competing in the ICCAD 2015 Contest. Finally, Section 3.4.3 brings evidence of the impact on clock-tree compactness.

3.4.1 Experimental Infrastructure

The ICCAD 2015 Contest infrastructure consists of 8 industrial circuits with sizes between 768k and 1.93M cells, a set of scripts, and a cell library with non-linear delay model (KIM et al., 2015). The remaining guidelines and metrics are the same as those employed in the old infrastructure (see Section B.1).

The clock routing estimation was improved to overcome the limitation of a single buffer driving the clock network. The infrastructure assumes a hierarchical clock distribution composed of global and local clock networks. The global network is responsible for delivering the clock signal from its source to the local clock networks and is assumed ideal (i.e. zero resistance). The local clock distributions are implemented as buffered trees of registers. Each buffered tree is constructed by clustering groups of closely placed registers, where each cluster is driven by a Local Clock Buffer (LCB). The local clock distribution is routed using FLUTE tool (CHU; WONG, 2008). Figure 33 illustrates the hierarchical clock distribution[3]. These improvements drastically reduced the large clock skews observed in the 2014 infrastructure to more realistic values.

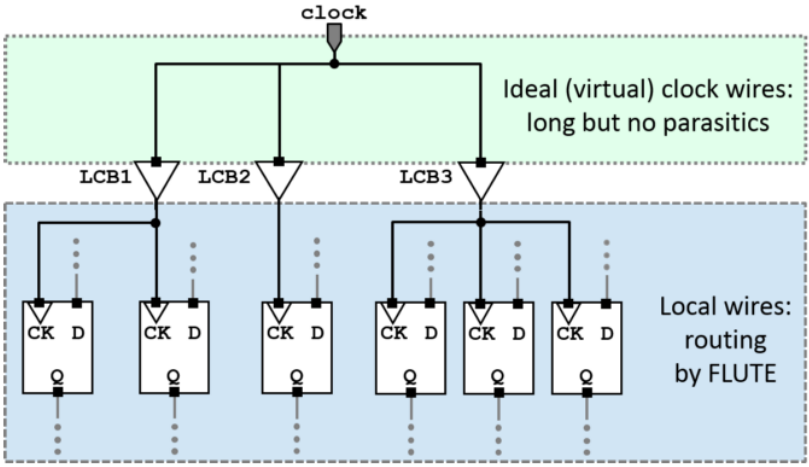


Figure 33: **ICCAD 2015 ITDP Contest's clock distribution.** *The global clock distribution is ideal, while the local distribution is routed using FLUTE. Obtained from (KIM et al., 2015).*

[3]The proposed IRP technique was applied only to the scope of local clock trees, i.e., the routing from LCBs to registers.

Table 2: **ICCAD 2015 ITDP Contest benchmark suite.**

Circuit	# of standard cells	# of sequential elements	# of macros	target clock period (ns)	target density	max. displ. limits (μm)
superblue18	768068	103544	653	7.0	0.85	30/400
superblue4	795645	176895	3471	6.0	0.90	50/500
superblue16	981559	142543	419	5.5	0.85	20/400
superblue5	1086888	114103	1872	9.0	0.85	30/400
superblue1	1209716	144266	3787	9.0	0.80	40/500
superblue3	1213253	167923	2074	10	0.87	40/400
superblue10	1876103	241267	1696	10	0.87	20/400
superblue7	1931639	270219	4910	5.5	0.90	50/400

Table 2 gives an overview of the ICCAD 2015 Contest benchmark suite, detailing the number of standard cells, number of registers, number of macro blocks, target clock period, target density (maximum is 1), and maximum displacement limits for short and long scenarios.

In the experiments, we adopted the following values for the predefined constants used in the algorithms: $max\_it = 20$, $\rho = 200$, $num\_samples = 5$. We observed experimentally that those values lead to a good tradeoff between solution quality and runtime. To set the bound $SWI_{max} = 4\%$, we adopted a value compatible with the signal wirelength degradation reported in related works (WANG et al., 2007; LEE; MARKOV, 2012). As opposed to most related works on register placement, which disregard density, we defined the bound $PDI_{max} = 0.5\%$ arbitrarily. However, neither signal wirelength nor placement density increase reached that limit for any of the tested circuits. To comply with the ICCAD Contest quality metric, where late violations are five times more important than early violations, we modified Algorithm 4 (line 21) by weighting 5:1. The initial values for all the Lagrange Multipliers were set to 1.

3.4.2 The Impact of the Proposed Technique

The proposed ITDP technique alone (i.e. without the IRP step from Section 3.3.3) was submitted to the ICCAD Contest 2015 and produced results that were awarded the first place in that contest, as detailed in Appendix C. That is why the experimental validation herein presented compares the proposed technique (i.e. IRP and ITDP) with those that obtained the 2^{nd} and 3^{rd} place in the ICCAD 2015 Contest.

Tables 3 and 4 display the results for short and long displacement

constraints, respectively. Since the major difference between the results presented in this section and in Appendix C lies on the IRP step[4], three distinct placements were evaluated under each metric: the (Initial) solution provided in the contest infrastructure, the highest quality solution among 2^{nd} and 3^{rd} place techniques in the contest (Best*), and the solution obtained from the proposed approach (Proposed). Columns 3-6 show the values obtained for late/early WNS and TNS, while columns 7 and 8 report ABU and quality metrics[5]. Additionally, columns 9 and 10 report the total clock and signal wirelengths. It is important to notice that the clock wirelength corresponds only to the local clock distribution, as the global distribution is assumed ideal. The last column displays the runtimes reported in Kim et al. (2015) and the runtimes we measured when running the proposed approach. The bottom row shows the average reduction obtained by the proposed approach when compared to Best*. A negative percentage indicates worsening from the perspective of a given metric. Let us analyze the results according to four distinct aspects: timing closure, signal wirelength and density, clock-tree compactness, and runtime.

Timing Closure: When focusing on late constraints under short displacement (Table 3), the proposed approach obtains average WNS and TNS reductions over the Best* solution around 0.8% and 6.7%, respectively, reaching up to 24% of TNS reduction on circuit *superblue*16. Under long displacement (Table 4), the average WNS and TNS reductions over the Best* solution are around 4% and 21.4%, respectively, reaching up to 56% of TNS reduction on circuit *superblue*16.

When handling early constraints under short displacement (Table 3), the proposed approach obtained average WNS and TNS reductions over the Best* solution around 19.6% and 37.4%, respectively. Under long displacement (Table 3), large worsenings were observed under both WNS and TNS metrics also due to large outliers, especially in circuits *superblue*5 and *superblue*1.

Clock-tree compactness: Under both short and long displacement constraints, the proposed approach leads to clock wirelength reductions around 15.7% and 17.2%, respectively, as compared to the Best* solution. Such improvement provides clear evidence that clock-tree awareness broadens the impact of incremental placement beyond conventional ITDP.

[4] A minor difference in the ITDP step is the increase of *max_it* parameter from 5 to 20.

[5] We adopted the same quality metric that was employed in the ICCAD 2015 contest, which casts both timing violation reduction and density penalty into a single number. For further details on this metric, please, refer to the Appendix C

Table 3: **Results for the ICCAD 2015 ITDP Contest benchmarks under short displacement constraints.** *Best* corresponds the the highest quality solution among 2^{nd} and 3^{rd} places in the contest.*

		Late		Early				Steiner WL		
Benchmark	**Solution**	**WNS** (ns)	**TNS** (μs)	**WNS** (ps)	**TNS** (ns)	**ABU** (10^{-2})	**Quality**	**Clock** (m)	**Signal** (m)	**Runtime** (min.)
superblue18	Initial	-4.55	-1.03	-19.01	-0.28	0.04	—	0.54	57.1	—
cells: 768k	Best*	-4.15	-1.00	-12.65	-0.07	0.04	257.7	0.54	57.1	678.5
macros: 0.6k	Proposed	-4.06	-0.93	-2.61	-0.03	0.01	434.1	0.44	58.0	5.0
superblue4	Initial	-6.22	-3.48	-12.55	-0.52	0.04	—	0.89	70.6	—
cells: 796k	Best*	-6.00	-3.27	-11.67	-0.27	0.04	179.0	0.89	70.6	718.0
macros: 3.4k	Proposed	-5.76	-3.03	-12.43	-0.05	0.03	351.1	0.76	71.3	7.2
superblue16	Initial	-4.58	-0.78	-10.65	-0.11	0.03	—	0.75	92.6	—
cells: 982k	Best*	-4.25	-0.73	-0.84	0.00	0.03	386.4	0.75	92.6	472.4
macros: 0.4k	Proposed	-4.37	-0.55	-0.07	0.00	0.02	616.3	0.66	93.2	6.7
superblue5	Initial	-25.70	-6.97	-36.77	-0.59	0.02	—	0.67	106.9	—
cells: 1.09M	Best*	-25.12	-6.90	-30.52	-0.26	0.02	148.2	0.67	106.9	717.6
macros: 1.8k	Proposed	-25.07	-6.79	-22.16	-0.27	0.02	185.8	0.55	107.5	6.7
superblue1	Initial	-4.98	-0.46	-9.34	-0.32	0.05	—	0.84	95.0	—
cells: 1.21M	Best*	-4.66	-0.39	-4.21	-0.03	0.05	410.9	0.84	95.1	718.4
macros: 3.7k	Proposed	-4.68	-0.38	-7.59	-0.01	0.02	432.4	0.72	95.7	8.2
superblue3	Initial	-10.15	-1.50	-78.36	-1.46	0.03	—	0.86	113.5	—
cells: 1.21M	Best*	-9.71	-1.45	-50.72	-0.56	0.03	214.2	0.86	113.5	718.4
macros: 2.0k	Proposed	-9.44	-1.38	-38.07	-0.34	0.02	328.6	0.73	114.5	8.7
superblue10	Initial	-16.49	-33.15	-8.62	-0.62	0.04	—	1.39	203.9	—
cells: 1.88M	Best*	-16.16	-32.99	-5.01	-0.34	0.04	145.9	1.39	203.9	719.8
macros: 1.6k	Proposed	-16.17	-32.59	-4.71	-0.44	0.02	132.6	1.09	205.6	11.7
superblue7	Initial	-15.22	-1.86	-7.65	-1.99	0.03	—	1.46	138.7	—
cells: 1.97M	Best*	-15.22	-1.75	-6.93	-1.93	0.03	70.3	1.48	138.7	579.3
macros: 4.9k	Proposed	-15.22	-1.70	-5.95	-1.87	0.02	121.4	1.32	139.0	14.3
Average Red. vs Best*		0.8%	6.7%	19.6%	37.4%	1.8%	46.5%	15.8%	-0.8%	—

Signal wirelength and density: In the proposed approach, the support to fostering routability consists in imposing bounds on the increase of placement density and signal wirelength. Therefore, the average variation in signal wirelength and in placement density (ABU) can be used to estimate the approach's potential impact on routability (VISWANATHAN et al., 2010). As compared to the related techniques, the proposed approach leads to ABU reductions around 2%, regardless of how tight displacement constraints are (either short or long). Although a slight worsening was observed in signal wirelength, it was kept below 1% of increase. Besides, under short and long displacement constraints, the proposed approach achieves quality metrics that are 46.5% and 72.7% higher than the Best* solution. These results provide a clear evidence that despite some worsenings observed under early violations for a few circuits, the proposed technique can provide better quality and shorter clock tree wirelength estimates than the related techniques.

Runtime: Due to the distinct configurations of the workstations where different techniques were run, no meaningful average reduction

Table 4: **Results for the ICCAD 2015 ITDP Contest benchmarks under long displacement constraints.** *Best* corresponds the the highest quality solution among 2^{nd} and 3^{rd} places in the contest.*

		Late		Early				Steiner WL		
Benchmark	**Solution**	**WNS** (ns)	**TNS** (μs)	**WNS** (ps)	**TNS** (ns)	**ABU** (10^{-2})	**Quality**	**Clock** (m)	**Signal** (m)	**Runtime** (min.)
superblue18	initial	-4.55	-1.03	-19.01	-0.28	0.04	—	0.54	57.1	—
cells: 768k	Best*	-3.73	-0.89	-6.01	-0.01	0.04	484.2	0.55	57.2	675.8
macros: 0.6k	Proposed	-3.71	-0.75	-0.97	0.00	0.01	675.9	0.45	58.1	7.0
superblue4	initial	-6.22	-3.48	-12.55	-0.52	0.04	—	0.89	70.6	—
cells: 796k	Best*	-6.22	-3.43	-9.37	-0.08	0.04	208.5	0.93	70.7	17.8
macros: 3.4k	Proposed	-5.52	-2.27	-14.78	-0.07	0.02	572.7	0.77	71.6	34.6
superblue16	initial	-4.58	-0.78	-10.65	-0.11	0.03	—	0.75	92.6	—
cells: 982k	Best*	-3.61	-0.62	-2.83	-0.01	0.03	558.8	0.80	92.6	719.4
macros: 0.4k	Proposed	-3.83	-0.27	-1.52	-0.01	0.02	1,021.1	0.68	93.5	6.4
superblue5	initial	-25.70	-6.97	-36.77	-0.59	0.02		0.67	106.9	—
cells: 1.09M	Best*	-25.69	-6.79	-9.40	-0.15	0.02	249.4	0.76	107.7	19.1
macros: 1.8k	Proposed	-23.88	-5.84	-25.57	-0.27	0.03	333.2	0.56	107.8	9.6
superblue1	initial	-4.98	-0.46	-9.34	-0.32	0.05	—	0.84	95.0	—
cells: 1.21M	Best*	-4.88	-0.45	-9.34	-0.12	0.05	163.7	0.86	95.2	40.3
macros: 3.7k	Proposed	-4.51	-0.33	-15.20	-0.03	0.02	452.7	0.73	95.9	57.8
superblue3	initial	-10.15	-1.50	-78.36	-1.46	0.03	—	0.86	113.5	—
cells: 1.21M	Best*	-9.27	-1.30	-23.23	-0.17	0.03	427.7	0.91	113.7	35.7
macros: 2.0k	Proposed	-8.46	-1.14	-38.93	-0.41	0.02	522.2	0.78	114.0	12.6
superblue10	initial	-16.49	-33.15	-8.62	-0.62	0.04	—	1.39	203.9	—
cells: 1.88M	Best*	-16.48	-33.12	-5.01	-0.03	0.04	231.8	1.42	204.1	48.7
macros: 1.6k	Proposed	-16.03	-30.90	-5.24	-0.44	0.02	184.2	1.12	205.4	12.1
superblue7	initial	-15.22	-1.86	-7.65	-1.99	0.03	—	1.46	138.7	—
cells: 1.97M	Best*	-15.22	-1.64	-6.93	-1.94	0.03	129.6	1.48	138.7	720.0
macros: 4.9k	Proposed	-15.22	-1.51	-5.95	-1.87	0.02	224.3	1.33	139.2	37.7
Average Red. vs Best*		4.0%	21.4%	-27.5%	-145.2%	1.83%	72.7%	17.2%	-0.7%	—

in runtime could be computed. Nevertheless, for the largest circuit (almost 2M cells), the proposed approach took 14 and 37 minutes under short and long displacement constraints, respectively.

Evidences of an effective ordering: The adopted order for the proposed decomposition assumed that few registers would be touched by ITDP after register placement. For the adopted infrastructure, on average, 14.2% of the standard cells correspond to registers, as shown Figure 34 (a). The experimental results (under long displacement) revealed that, on average, incremental register placement moves 31% of standard cells (17.7% + 13.3%), as shown in Figure 34 (b). Such large number of moved combinational cells is a side effect of the legalization step (since combinational cells must leave space for registers) and of the iterative detailed placement that fosters routability. Figures 34 (c) and (d) show that, since incremental ITDP only relocates cells belonging to the critical paths, it moves 1.8% of the cells and only 0.2% of them are registers. This practical evidence confirms the initial assumption, meaning that the proposed approach adopted adequate order for coupling the subproblems.

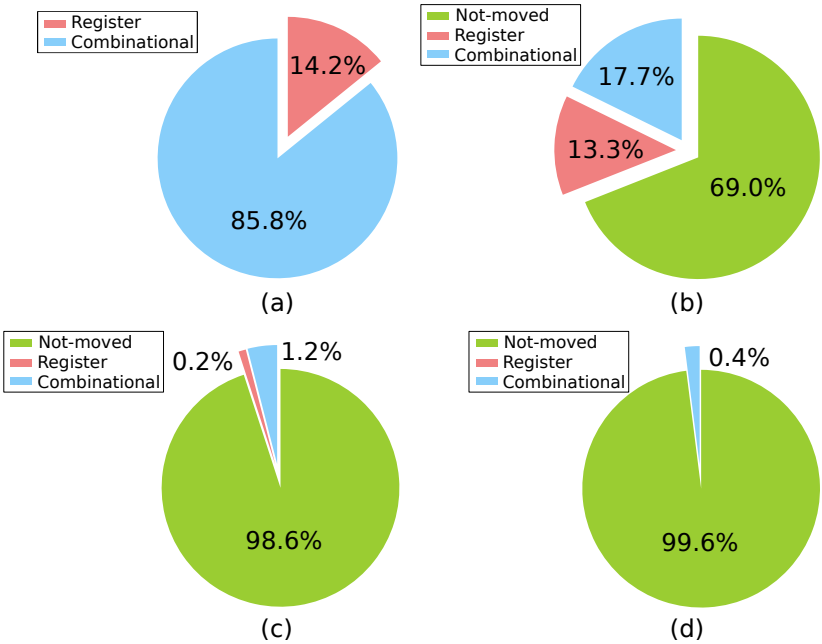


Figure 34: **Statistics on all ICCAD 2015 ITDP Contest's Circuits.** *(a) Percentage of registers and combinational among all standard cells. (b) Percentage of registers, combinational, and non-moved cell after incremental register placement, (c) after Solving LR, and (d) and after non-critical cell relocatin.*

Impact of each subproblem: After detailing the results of the proposed joint approach, we illustrate the contribution of each of the coupled techniques. Table 5 shows the intermediate results after incremental register placement (IRP), after the Solving LR technique (LR), and the final results after Non-Critical Cell Relocation (NCR). Notice that, due to the more realistic setup of clock network, the reductions of late timing violations obtained by the ITDP technique (LR + NCR) are prominent. Besides, the IRP improvement on clock wirelength does not induce large reductions on clock skew and thus has a very small impact on late timing violations. Since the clock skew in the initial solutions represents, on average, only 0.64% of the target clock period, which is reasonable for a realistic scenario (LEE; KIM; MARKOV, 2010), the skew affects much more early violations than late ones. This comes from the fact that early violations are related to short paths and thus more sensitive to small skew variations. This can be readily observed from the columns reporting the late scenario, as these reductions were

obtained only during the LR and the NCR steps. On the other hand, most of the reductions on early violations came from the IRP step.

Table 5: **Results for the ICCAD 2015 ITDP Contest benchmarks after solving each subproblem.**

		Late		Early				Steiner WL		
Benchmark	**Solution**	**WNS** (ns)	**TNS** (ns)	**WNS** (ns)	**TNS** (ns)	**ABU** (10^{-2})	**Quality**	**Clock** (m)	**Signal** (m)	**Runtime** (min)
superblue18	IRP	-4.61	-1.04	-1.75	-0.01	0.01	276.2	0.44	58.0	2.4
short	LR	-4.06	-0.93	-2.61	-0.03	0.01	434.1	0.44	58.0	4.6
	NCR	-4.06	-0.93	-2.61	-0.03	0.01	434.1	0.44	58.0	5.0
superblue4	IRP	-6.23	-3.51	-12.43	-0.05	0.03	174.0	0.76	71.2	3.1
short	LR	-5.77	-3.06	-12.43	-0.05	0.03	343.7	0.76	71.2	6.4
	NCR	-5.76	-3.03	-12.43	-0.05	0.03	351.1	0.76	71.3	7.2
superblue16	IRP	-4.61	-0.83	-2.73	-2.73E-3	0.01	207.4	0.65	93.1	1.9
short	LR	-4.37	-0.57	-0.07	-6.99E-5	0.02	601.0	0.66	93.2	5.3
	NCR	-4.37	-0.55	-0.07	-6.99E-5	0.02	616.3	0.66	93.2	6.7
superblue5	IRP	-25.70	-6.99	-21.39	-0.24	0.02	155.7	0.54	107.9	3.4
short	LR	-25.07	-6.79	-22.16	-0.27	0.02	185.8	0.55	107.5	5.7
	NCR	-25.07	-6.79	-22.16	-0.27	0.02	185.8	0.55	107.5	6.7
superblue1	IRP	-4.98	-0.46	-7.56	-0.02	0.01	214.7	0.72	95.6	5.0
short	LR	-4.68	-0.38	-7.59	-0.01	0.02	428.4	0.72	95.6	7.2
	NCR	-4.68	-0.38	-7.59	-0.01	0.02	432.4	0.72	95.7	8.2
superblue3	IRP	-10.17	-1.51	-38.07	-0.33	0.02	201.6	0.72	114.5	4.9
short	LR	-9.50	-1.40	-38.07	-0.34	0.02	312.3	0.73	114.5	7.9
	NCR	-9.44	-1.38	-38.07	-0.34	0.02	328.6	0.73	114.5	8.7
superblue10	IRP	-16.55	-33.28	-4.71	-0.43	0.02	103.9	1.09	205.6	7.1
short	LR	-16.17	-32.82	-4.71	-0.44	0.02	125.6	1.09	205.6	10.4
	NCR	-16.17	-32.59	-4.71	-0.44	0.02	132.6	1.09	205.6	11.7
superblue7	IRP	-15.22	-1.86	-5.95	-1.87	0.01	31.6	1.32	139.0	9.2
short	LR	-15.22	-1.75	-5.95	-1.87	0.02	93.4	1.32	139.0	13.0
	NCR	-15.22	-1.70	-5.95	-1.87	0.02	121.4	1.32	139.0	14.3
superblue18	IRP	-4.58	-1.04	-0.37	-1.01E-3	0.01	297.0	0.45	58.0	3.9
long	LR	-3.73	-0.77	-0.97	-2.33E-3	0.01	653.1	0.45	58.0	6.5
	NCR	-3.71	-0.75	-0.97	-2.33E-3	0.01	675.9	0.45	58.1	7.0
superblue4	IRP	-6.22	-3.53	-14.78	-0.05	0.02	152.0	0.75	71.5	29.1
long	LR	-5.52	-2.31	-14.78	-0.07	0.02	560.0	0.77	71.5	33.7
	NCR	-5.52	-2.27	-14.78	-0.07	0.02	572.7	0.77	71.6	34.6
superblue16	IRP	-4.61	-0.81	-1.52	-4.07E-3	0.01	235.5	0.65	93.2	2.1
long	LR	-3.83	-0.27	-1.52	-5.23E-3	0.02	1,021.1	0.68	93.5	5.1
	NCR	-3.83	-0.27	-1.52	-5.23E-3	0.02	1,021.1	0.68	93.5	6.4
superblue5	IRP	-25.72	-6.99	-25.57	-0.26	0.03	136.6	0.54	107.7	5.7
long	LR	-23.91	-5.88	-25.57	-0.27	0.03	326.6	0.56	107.8	8.5
	NCR	-23.88	-5.84	-25.57	-0.27	0.03	333.2	0.56	107.8	9.6
superblue1	IRP	-4.99	-0.46	-7.56	-0.01	0.01	214.4	0.72	95.8	54.9
long	LR	-4.53	-0.34	-15.20	-0.03	0.02	441.9	0.73	95.9	57.2
	NCR	-4.51	-0.33	-15.20	-0.03	0.02	452.7	0.73	95.9	57.8
superblue3	IRP	-10.16	-1.51	-38.93	-0.39	0.02	195.9	0.77	113.9	7.9
long	LR	-8.65	-1.18	-38.93	-0.41	0.02	485.4	0.78	114.0	11.8
	NCR	-8.46	-1.14	-38.93	-0.41	0.02	522.2	0.78	114.0	12.6
superblue10	IRP	-16.52	-33.27	-5.24	-0.44	0.02	94.0	1.10	205.3	7.4
long	LR	-16.04	-31.82	-5.24	-0.44	0.02	155.9	1.12	205.3	10.8
	NCR	-16.03	-30.90	-5.24	-0.44	0.02	184.2	1.12	205.4	12.1
superblue7	IRP	-15.22	-1.87	-5.95	-1.87	0.01	30.4	1.32	139.1	32.4
long	LR	-15.22	-1.59	-5.95	-1.87	0.02	183.2	1.33	139.1	36.3
	NCR	-15.22	-1.51	-5.95	-1.87	0.02	224.3	1.33	139.2	37.7

3.4.3 Further Evidences of Clock-Tree Compactness

Figure 35 compares the local clock tree routing before and after applying the proposed technique for two circuits from the ICCAD 2015 Contest under short displacement constraints. The circuits are sorted by number of cells to show how the proposed technique behaves for different circuit sizes. To comply with the contest infrastructure, the local clock tree routing was estimated using FLUTE (CHU; WONG, 2008). It can be seen that the proposed technique efficiently compacts the clock tree, achieving up to 17% shorter wirelength. Observe that the impact on routability is quite small since the placement density is reduced for all circuits while the signal wirelength degradations are less than 1%. This is a consequence of the detailed placement step applied after each IRP iteration.

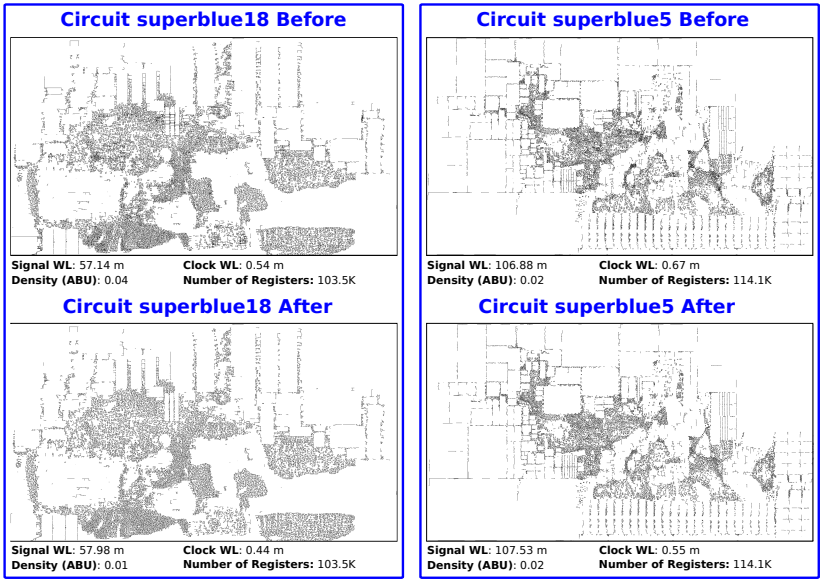


Figure 35: **Snapshots of the clock routing for ICCAD 2015 circuits.** *Local clock trees for two circuits from the ICCAD 2015 Contest before (above) and after (below) applying the proposed incremental register placement technique.*

3.5 CONCLUSIONS

About 31% of the standard cells were relocated after incremental register placement, roughly 1.5% of them were touched by TDP, and less than 0.2% are registers[6]. This indicates that the proposed approach adopted an adequate order for coupling the optimizations. To obtain reductions around 16% in clock wirelength, the proposed approach incurred a penalty of less than 1% in signal wirelength and no penalty at all on placement density. Besides, the proposed technique showed consistent reductions in timing violations. Thus, there is strong experimental evidence that the effectiveness on timing closure and clock-tree compactness was not obtained at the expense of a less routable circuit.

[6] Results considering the ICCAD Contest 2015 infrastructure.

4 INCREMENTAL TIMING-DRIVEN LAYER ASSIGNMENT

The increasing impact of interconnect delay on the overall circuit performance represents a bottleneck for timing closure. The worse scaling of interconnect delay, as compared to cell delay, is a consequence of the quadratic increase of wire resistance per unit length (WEI et al., 2013). Such scenario has shifted the research spotlight towards efficient interconnect synthesis and timing optimization techniques like buffer insertion, timing-driven placement, and layer assignment.

Modern technologies may provide twelve or more metal layers with different widths and thicknesses (ALPERT et al., 2010), where upper layers are wider and thicker than lower ones. Albeit their resistance is reduced quadratically, the upper layers require more area and therefore offer less resources for routing. This raises the importance of incremental timing-driven layer assignment techniques, which must properly re-assign critical interconnect segments to upper layers in order to improve the overall circuit timing, no matter how global routing was performed (either through 3D or 2D routers, whether timing-aware or not) (ALPERT et al., 2010; HU; LI; ALPERT, 2009; YU et al., 2015).

Although the choice of a proper timing engine is crucial, related works have relied on simplified models for interconnect capacitance (lumped capacitance) and delay (Elmore's model). Such models are pessimistic because they ignore second order effects (like resistive shielding), which become prominent in the face of multiple metal layers with very different electrical characteristics. Being overly pessimistic, they end up hindering timing closure and resulting in over-allocation of resources (such as vias) (PURI; KUNG; DRUMM, 2002; KAHNG et al., 2013a; REIMANN; SZE; REIS, 2016).

Albeit accurate timing engines (e.g. signoff timing analyzers) are available from conventional EDA packages, they have not been exploited for layer assignment because industrial engines (to preserve intellectual property) do not report timing information for (inner) net segments, but only for cell pins and timing endpoints. The techniques proposed so far seem to take such opacity for a barrier and keep relying on inaccurate built-in engines (DONG; AO; LUO, 2015; LIU et al., 2016).

Another limitation of such techniques lies in the inaccurate objective function (SAXENA; LIU, 2001; DONG; AO; LUO, 2015; YU et al., 2015). Since they minimize the sum of net delays, which might not

lead to timing improvements in the critical paths, they further hinder timing closure.

This chapter presents a novel incremental layer assignment technique that overcomes such limitations of previous works. It not only handles critical and non-critical net segments simultaneously, but also exploits flow conservation conditions to extract information for each net segment individually, thereby enabling the use of an external signoff timing engine. The main contributions of this chapter are:

- A binary integer programming formulation for incremental layer assignment targeting at total negative slack optimization while modeling each net segment separately.
- A cast of the binary integer programming into a Lagrangian Relaxation formulation that exploits flow conservation conditions to decouple the layer assignment technique from the timing analysis engine.
- A min-cost network flow technique (to solve the Lagrangian Relaxation formulation) that independently models each net segment while capturing the impact of capacitance and slew variation on neighboring segments and cells. Besides, an efficient edge-pruning methodology reduces runtime and via count.
- A strategy to exploit the slacks reported by a signoff timer to obtain accurate Lagrange Multipliers for net segments. A detailed analysis of the impact of the timing engine accuracy in the proposed technique is also presented.

The impact of such contributions on timing closure was experimentally compared with two state-of-the-art techniques (YU et al., 2015; LIU et al., 2016) for circuits derived from those available within the ICCAD 2015 Contest infrastructure (KIM et al., 2015). A signoff analyzer was used as a golden timing engine to evaluate the final 3D routing solutions obtained after the application of each of the three incremental layer assignment techniques under comparison. The proposed technique resulted in 50% less timing violations (under total negative slack metric) while using a similar number of vias, as compared to the best results obtained from the related works.

The remaining of this chapter is organized as follows. Section 4.1 reviews the state-of-the-art of timing-driven layer assignment while Section 4.2 details the adopted timing modeling and the problem definition. Then Section 4.3 presents the proposed mathematical formulation for the target problem. Section 4.4 describes the proposed technique.

Section 4.5 details its experimental evaluation. Finally, Section 4.6 draws the main conclusion.

4.1 RELATED WORK

Global routing can be accomplished either by direct 3D routing (native layer assignment) (WU; DAVOODI; LINDEROTH, 2011; HELD et al., 2015) or through 2D routing followed by a layer assignment step (CHO et al., 2007; LIU et al., 2013). *Incremental* layer assignment plays the important role of improving global routing and it can target different objectives, such as via count minimization (LIU; LI, 2011), antenna alleviation (WU; HU; MAHAPATRA, 2005; DONG; AO; LUO, 2015), and timing optimization (YU et al., 2015; LIU et al., 2016), the latter being the focus of this chapter. That is why this section addresses only related works on incremental layer assignment for timing optimization.

Most methods addressing incremental layer assignment rely on a 3D routing grid that is defined by parallel routing planes (*layers*), which are divided into rectangular cells, called *G-cells*, as illustrated in Figure 36(a). The boundaries between adjacent cells on the same plane are associated with intra-plane routing tracks. Two *G-cells* from different but consecutive planes are interconnected through vias. As a result, most layer assignment techniques model the routing grid as a graph whose vertices represent *G-cells* and whose edges represent the connectivity between two adjacent *G-cells*, as shown in Figure 36(b). Capacities are associated with the edges (to capture routing constraints) and net pins are associated with vertices.

Several techniques perform *net-by-net* iterative improvement steps to accomplish the overall timing optimization. **Saxena and Liu (2001)** proposed a fast iterative heuristic to minimize the worst net delays. It relies on the notion of area quota to mimic edge capacities. In every iteration, the area quota of each net is kept proportional to its Manhattan wirelength. **Li et al. (2008)** presented an interconnect synthesis technique for simultaneous buffer insertion and layer assignment when targeting slew and net delay recovery. The authors extended the classic Van Ginneken's dynamic programming algorithm to accommodate new pruning strategies. **Hu, Li and Alpert (2008)** proved that layer assignment under timing constraints is NP-complete and devised a polynomial time approximation scheme. The authors proposed a fast binary search technique that queries a dynamic programming oracle

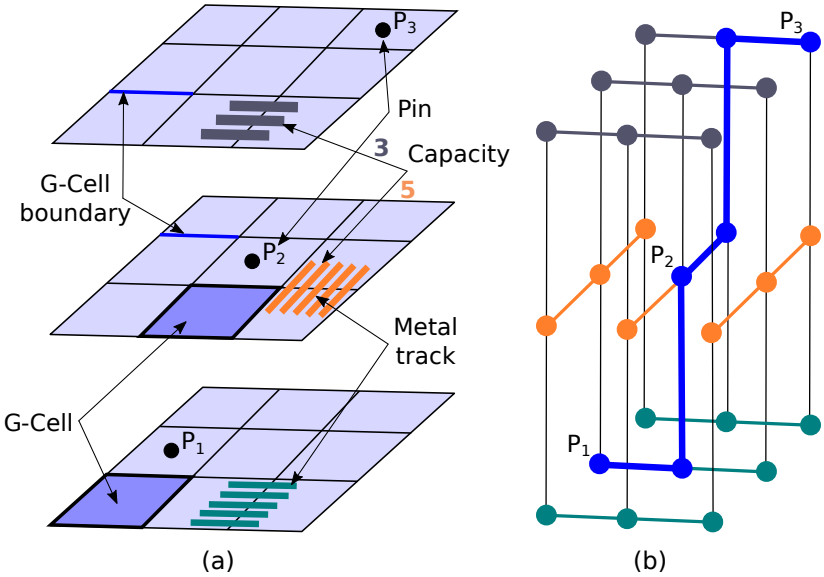


Figure 36: **3D global routing grid example.** *(a) 3D global routing grid with 3 layers and 9 G-cells each. (b) Corresponding grid graph and 3D routing for a 3-pin net.*

about lower and upper-bound solutions. Later on, **Hu, Li and Alpert (2009)** proposed a new polynomial algorithm to improve the theoretical complexity derived in their previous work (HU; LI; ALPERT, 2008). The authors revisited some limitations of the dynamic programming oracle and proposed a linear-time algorithm. **Dong, Ao and Luo (2015)** combined dynamic programming and negotiation strategies to minimize the maximum net delay with low overhead in via count. The main limitation of all such techniques results exactly from their net-by-net approach, which may lead to locally-optimal solutions, as highlighted in (YU et al., 2015). The very limited availability of wide and thick wires may lead to poor timing optimization when an inadequate net ordering is adopted. Besides, some of those techniques assume that all segments of a given net must share the same layer, which may induce over-allocation.

A few techniques perform all-net simultaneous optimization. **Yu et al. (2015)** observed the limitations of net-by-net strategies and proposed a min-cost flow technique to simultaneously minimize the sum of net segment and via delays. To handle via capacity constraints, the authors devised a Lagrangian Relaxation formulation that incorporated those capacity constraints into the objective function. Therefore,

the employed min-cost flow modeling was able to handle simultaneously net segment delay, edge capacity, and via overflow. Recently, **Liu et al. (2016)** revisited some of the limitations from (YU et al., 2015) and proposed a semidefinite programming formulation to handle quadratic constraints, which are used to model via delay and via capacity. Their proposed framework targets nets belonging to the critical path and employs a self-adaptive algorithm to balance the distribution of nets among different threads. Unfortunately, the objective functions adopted in such works, namely the sum of net delays or the maximum net delay, turns out limiting potential improvements. Since large net delays might not lead to a timing violation at a given timing endpoint, those objective functions may end up inducing improvements in paths that do not impair timing closure, as illustrated in Figure 37. In addition, a net-delay-driven strategy is unaware of different setup constraints at sequential elements, which is essential to identify paths with negative slack. Figure 38 shows an example of how a net-delay-driven technique may not be effective to reduce violations. Although the net-delay-driven technique (Figure 38(a) and (b)) leads to a better compression of the net delay histogram (as expected), this does not translate into an actual compression of the negative slack histogram at the circuit's timing endpoints (Figs. 38(b) and (d)).

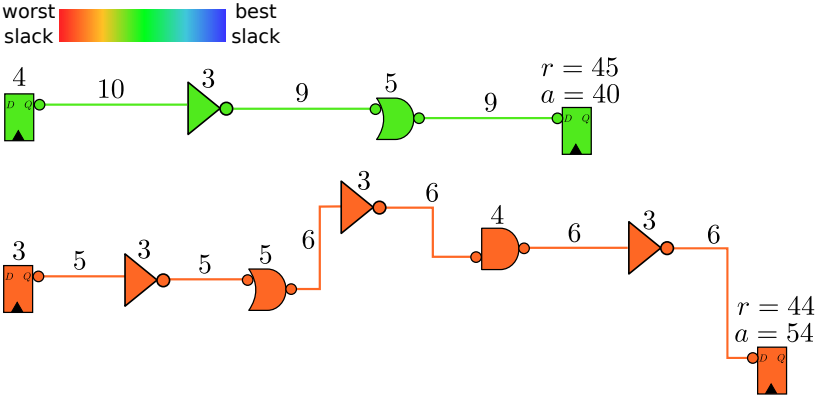


Figure 37: **Impact of net delay on timing violations.** *Comparison between a non-critical (top) and a critical path (bottom). Cells and wires are labeled with their delays. Required (r) and arrival (a) times are indicated at timing endpoints. Note that the non-critical path has the largest net delays, whereas the critical path has the lowest ones. Thus, net-delay-driven approaches may guide optimization off critical paths.*

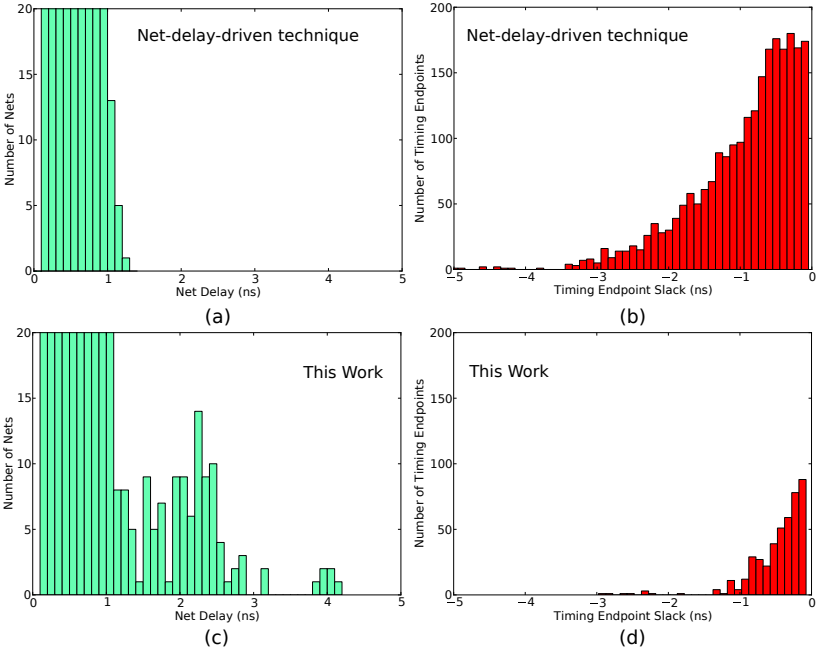


Figure 38: **Net delay and slack histograms.** *Net delay histograms (left) and timing endpoint slack histograms (right) for circuit superblue16. (a) and (b) show the behavior of the net-delay-driven technique (LIU et al., 2016) while (c) and (d) illustrate the impact of the technique proposed in this work.*

Most importantly, the main limitation of all incremental layer assignment techniques reported so far lies in the simplified timing model adopted to guide the optimization. The mismatch between the estimated and the actual timing is likely to hinder timing closure, as illustrated in Figure 39. Note that the simplified model overestimates WNS in all cases but one, the mismatch ranging from 5% to 40%. It also overestimates TNS in all cases, the mismatch ranging from 20% to almost 400%. Despite the clear inadequacy of overly pessimistic engines, the accurate signoff timing analyzers available from conventional EDA packages were never used by any technique reported so far. We put this down to the opacity of such analyzers, which report timing only for cell pins and timing endpoints, but not for inner net segments. Apparently, previous techniques took such opacity as an insurmountable barrier to the use of signoff timers during optimization. On the contrary, we realized that the key to overcoming their lack of

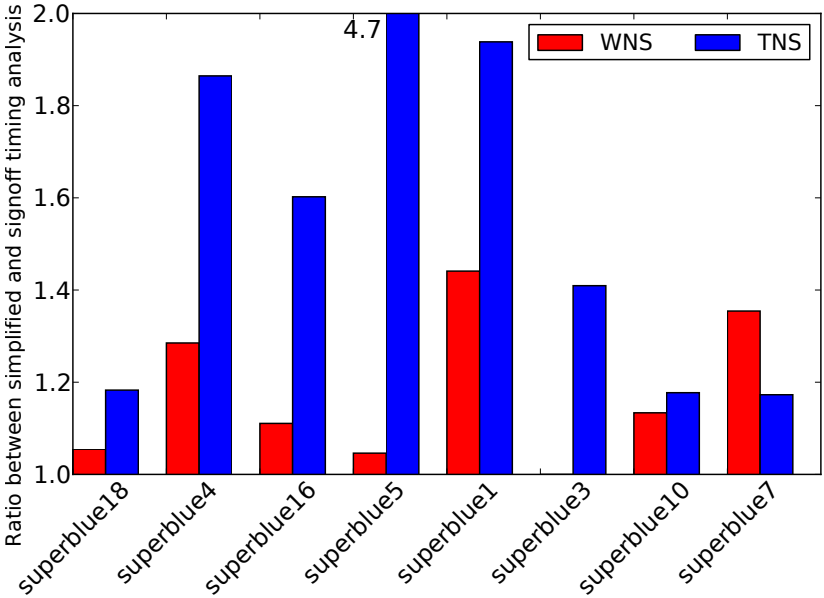


Figure 39: **Mismatch between simplified and signoff timing engine.** *Ratio between slack values obtained from a simplified timing engine (lumped capacitance and Elmore's delay) and from an industrial signoff timing engine. The over-estimation is shown for two metrics: Worst negative slack (WNS) and total negative slack (TNS).*

inner observability is the exploitation of flow conservation conditions so as to extract inner timing information. This motivated us to decouple incremental layer assignment from timing analysis and to exploit flow conservation conditions for enabling the use of an accurate signoff analyzer during optimization, as described in the next section.

4.2 PROBLEM DEFINITION

This section discusses the required background and presents the problem definition. First, Sections 4.2.1 and 4.2.2 detail the adopted routing grid and timing modeling. Finally, Section 4.2.3 defines the target problem tackled in this thesis.

4.2.1 Routing Grid Modeling

The layer assignment problem is usually defined over a 3D routing grid, as already illustrated in Figure 36. The routing grid can be modeled as a graph $G = (V, E)$, where each vertex represents a *G-cell* and each edge represents the connectivity between two adjacent *G-cells*. The set of edges is a partition $E = E^w + E^v$, where E^w is the set of edges induced by the boundaries between *G-cells* in the same plane and E^v is the set of edges induced by vias. Each edge in E^w has a capacity that represents the number of detailed routing tracks allowed to pass through that edge. Therefore, assuming an initial 3D global routing solution, the incremental layer assignment problem can be stated as follows: given a set $\mathcal{S}$ of net segments and a set $\mathcal{L}$ of routing layers, re-assign the segment layers in order to optimize some objective (for instance, minimize the number of vias (LIU; LI, 2011) or the number of timing violations). This work focuses on incremental layer assignment for reducing timing violations.

4.2.2 Timing Modeling

Although the basics of the adopted design representation and timing modeling were already introduced in Chapter 2, some concepts are repeated here for clarity. Besides, a few symbols introduced in Chapter 2 were slightly modified to make the problem formulation more clear. Since the problem tackled in this chapter targets late timing constraints, the superscript symbol (^{L}) indicating the late scenario was dropped for simplicity.

A sequential circuit can be represented by a set $\mathcal{C}$ of standard cells, a set $\mathcal{TE}$ of timing endpoints, and a set $\mathcal{TS}$ of timing startpoints (REN et al., 2007). The set $\mathcal{TE}$ includes both circuit output pads and register input pins, while the set $\mathcal{TS}$ includes both circuit input pads and register output pins. There is also a set $\mathcal{N}$ of nets representing the interconnections between these elements. Circuit arrival times are measured at input/output pins of each cell $c_j \in \mathcal{C}$ and at each $j \in (\mathcal{TE} \cup \mathcal{TS})$. The arrival time, denoted as a_j, corresponds to the latest time when a signal transition reaches a given timing point. The required time, denoted as r_j, corresponds to the latest time when the signal transition must reach each $j \in \mathcal{TE}$ to ensure the target clock frequency (KAHNG et al., 2011). To evaluate how far a design is from timing closure, slacks are tracked at circuit timing endpoints as:

$slk_j = r_j - a_j, \forall j \in \mathcal{TE}$. Timing optimization techniques, like timing-driven placement (GUTH; LIVRAMENTO et al., 2015) and gate sizing (OZDAL; BURNS; HU, 2012), typically employ the total negative slack (TNS) metric to capture all timing endpoints with timing violations as follows: $\sum_{j \in \mathcal{TE}} \min(0, slk_j)$.

4.2.3 The Target Problem

Based on the previous discussions on routing grid and timing modeling, the proposed timing-driven layer assignment problem can be defined as follows: *Given an initial 3D routing solution, a set of net segments, and a routing grid with edge capacities for each layer, re-assign a subset of segments so as to minimize the circuit total late negative slack while satisfying edge capacity constraints.*

4.3 PROPOSED PROBLEM FORMULATION

This section discusses the proposed mathematical formulation for the target problem introduced in the previous section. Section 4.3.1 presents the proposed binary integer programming formulation for incremental layer assignment. Next, Section 4.3.2 shows how we cast that problem into a Lagrangian Relaxation formulation which is decoupled from timing analysis. Finally, Section 4.3.3 details how to obtain a Lagrange Multiplier for each net segment.

4.3.1 Proposed Binary Integer Programming Formulation

In order to formulate incremental layer assignment as a minimization problem, let us first define negative slack as $slk'_j = \min(0, slk_j)$ to ensure that only non-positive values are accounted for in the objective function. Therefore, the adopted objective function, defined in (4.1), aims to minimize the total negative slack. The inequality constraints (4.2) and (4.3) are introduced to model the negative slack variable slk'_j used in the objective function.

$$\textbf{Minimize} \quad : \quad -\sum_{j \in \mathcal{TE}} slk'_j \tag{4.1}$$

$$\textbf{Subject to} \quad : \quad slk'_j \leq 0, \ \forall j \in \mathcal{TE} \tag{4.2}$$

$$: \quad slk'_j \leq r_j - a_j, \ \forall j \in \mathcal{TE} \tag{4.3}$$

To obtain the arrival times at timing endpoints, a proper timing modeling is required for cells and interconnects. Let us first introduce the adopted modeling before presenting arrival time definitions. The cell delay and slew for each input/output pin-pair are represented through a non-linear delay model, which is taken from a standard cell library. Therefore, the delay of a given cell $c_j \in \mathcal{C}$ from an input pin i to its output pin is a function of the cell's input slew (σ_i) and cell's downstream capacitance (C_j^{down}), being computed as $\delta_{i,j}^c = LUT\_D(C_j^{down}, \sigma_i)$, where $LUT\_D(C_j^{down}, \sigma_i)$ is a function whose value is obtained by a query into the cell lookup table (BHASKER; CHADHA, 2009). The cell output slew is computed similarly.

Interconnections are modeled as RC trees, wherein each net segment is defined as a π-model[1]. The delay of a given net segment $s_j \in \mathcal{S}$ assigned to a layer $l_q \in \mathcal{L}$ is denoted as $\delta_j^s(q)$. Similar to net segments, each via is modeled as an RC π-model. In this way, the via delay between two net segments $s_i, s_j \in \mathcal{S}$ at layers $l_p, l_q \in \mathcal{L}$ is computed as $\delta_{i,j}^v(p,q) = \sum_{k=p}^{q-1} \delta^v(k)$, where $\delta^v(k)$ corresponds to the via delay between two consecutive layers l_k and l_{k+1}. Before introducing the arrival time modeling, let us define in Equation (4.4) a binary decision variable for each net segment. The constraint (4.5) ensures that each segment is assigned to one and only one layer.

$$\alpha_{j,q} = \begin{cases} 1, & \text{if } s_j \in \mathcal{S} \text{ is assigned to } l_q \in \mathcal{L} \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

$$\sum_{l_q \in \mathcal{L}} \alpha_{j,q} = 1, \ \forall s_j \in \mathcal{S} \tag{4.5}$$

Let $\mathcal{I}_i^c$ denote the set of indices to each input pin of $c_j \in \mathcal{C}$. Therefore, the arrival time of cell c_j can be modeled as in (4.6). Observe that $\delta_{i,j}^v$ captures the delay of vias connecting the cell input pin i to its

[1] A single π per segment is by no means restrictive since the proposed technique supports any number of π's per segment.

corresponding net segment. The delay of via $\delta^v_{i,j}$ depends on the layer assigned to the segment connected to the input pin i of c_j and the layer of the input pin itself. Therefore, $\delta^v_{i,j}$ serves as a shorthand notation for Equation (4.7). For simplicity, this equation assumes that the cell pin is routed in the first layer.

$$a_i + \delta^v_{i,j} + \delta^c_{i,j} \leq a_j, \forall i \in \mathcal{I}^c_j, \text{and } \forall c_j \in \mathcal{C} \tag{4.6}$$

$$\delta^v_{i,j} = \sum_{l_p \in \mathcal{L}} \alpha_{i,p} \cdot \sum_{k=1}^{p-1} \delta^v(k) \tag{4.7}$$

Typically, timing optimization techniques model the net arrival times only at source and sink pins. Unfortunately, such kind of net modeling is less flexible and therefore more appropriate for steps before global routing, when net segment information is not yet available (OZDAL; BURNS; HU, 2012; LIVRAMENTO et al., 2014; REIMANN; SZE; REIS, 2016). Differently, we propose to split the net arrival times into a finer segment granularity, as defined in Equation (4.8), where $\mathcal{I}^s_j$ denotes the set containing the index to either a segment or a pin connected to the input of $s_j \in \mathcal{S}$. Observe that $\delta^v_{i,j}$ captures the delay of vias connecting two consecutive segments s_i and s_j and depends on their assigned layers. Therefore, $\delta^v_{i,j}$ serves as a shorthand notation for Equation (4.9). The delay of segment δ^s_j depends on the layer assigned to segment s_j and thus serves as a shorthand notation for Equation (4.10). Figure 40 gives a small example of cell and segment arrival time modeling.

$$a_i + \delta^v_{i,j} + \delta^s_j \leq a_j, \forall i \in \mathcal{I}^s_j, \text{and } \forall s_j \in \mathcal{S} \tag{4.8}$$

$$\delta^v_{i,j} = \sum_{l_p \in \mathcal{L}} \sum_{l_q \in \mathcal{L}} \alpha_{i,p} \cdot \alpha_{j,q} \cdot \sum_{k=p}^{q-1} \delta^v(k) \tag{4.9}$$

$$\delta^s_j = \sum_{l_q \in \mathcal{L}} \delta^s_j(q) \cdot \alpha_{j,q} \tag{4.10}$$

Finally, the constraint (4.11) ensures that the edge routing capacity between two adjacent *G-cells* in the same layer is not exceeded, where $\mathcal{R}^k_i$ denotes the set of indices to each net segment routed through the edge e_i on layer l_q.

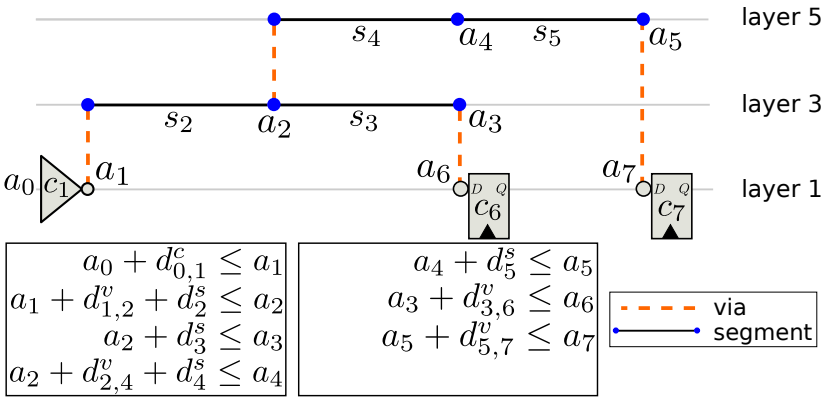


Figure 40: **Cell and net segment modelling.** *Circuit example with cell and net segment arrival time modeling for horizontal 3 layers. Each cell and segment is labeled with an index i and an arrival time a_i. Observe that the arrival time at segment s_2 is used in the inequalities that model the arrival times at segments s_3 and s_4.*

$$\sum_{j \in \mathcal{R}_i^k} \alpha_{j,q} \leq c^e_{i,q}, \ \forall e_i \in E^w, \text{and } \forall l_q \in \mathcal{L} \tag{4.11}$$

4.3.2 Proposed Lagrangian Relaxation Formulation

As highlighted in (OZDAL; BURNS; HU, 2012), incorporating timing analysis modeling directly into the optimization engine is not adequate due to the complexity of timing models adopted by modern timing engines. Therefore, we propose a Lagrangian Relaxation (LR) reformulation for the problem introduced in the previous subsection. It decouples the optimization engine from the timing analysis tool. The proposed LR formulation has two key differences with respect to well-known formulations adopted by gate sizing techniques found in the literature (CHEN; CHU; WONG, 1999; TENNAKOON; SECHEN, 2002; OZDAL; BURNS; HU, 2012; LIVRAMENTO et al., 2014; REIMANN; SZE; REIS, 2016): 1) we explicitly model the arrival times at each net segment, resulting into a finer granularity as compared to the conventionally modeling„ 2) we show how to take advantage of flow conservation conditions to obtain LMs for each net segment individually, thereby providing a better guidance towards layer re-assignment.

Therefore, we propose to relax the constraints that model the circuit timing information, i.e. Equations (4.2), (4.3), (4.6), and (4.8), and reflect them into the objective function. The inequalities modeling negative slacks at timing endpoints are accompanied by non-negative LMs denoted as λ'_j and λ_j. The remaining inequalities are multiplied by non-negative LMs denoted as $\lambda^c_{i,j}$ and λ^s_j for cell timing arcs and net segments, respectively. This leads to the following relaxed objective function:

$$\begin{aligned} L_\lambda : \quad & - \sum_{j \in \mathcal{TE}} slk'_j + \sum_{j \in \mathcal{TE}} \lambda'_j slk'_j + \sum_{j \in \mathcal{TE}} \lambda_j (slk'_j - r_j + a_j) \\ & + \sum_{c_j \in \mathcal{C}} \Big(\sum_{i \in \mathcal{I}^c_j} \lambda^c_{i,j} (a_i + \delta^v_{i,j} + \delta^c_{i,j} - a_j) \Big) \\ & + \sum_{s_j \in \mathcal{S}} \Big(\sum_{i \in \mathcal{I}^s_j} \lambda^s_j (a_i + \delta^v_{i,j} + \delta^s_j - a_j) \Big) \end{aligned} \tag{4.12}$$

For the subcircuit example illustrated in Figure 40, the Lagrangian Function obtained in Equation (4.12) would be stated as follows:

$$\begin{aligned} L_\lambda \quad : \quad & -slk'_6 - slk'_7 + \lambda'_6 slk'_6 + \lambda'_7 slk'_7 \\ & + \lambda_6 (slk'_6 - r_6 + a_6) + \lambda_7 (slk'_7 - r_7 + a_7) \\ & + \lambda^c_{0,1} (a_0 + \delta^c_{0,1} - a_1) + \lambda^s_2 (a_1 + \delta^v_{1,2} + \delta^s_2 - a_2) \\ & + \lambda^s_3 (a_2 + \delta^s_3 - a_3) + \lambda^s_4 (a_2 + \delta^v_{2,4} + \delta^s_4 - a_4) \\ & + \lambda^s_5 (a_4 + \delta^s_5 - a_5) + \lambda^c_6 (a_3 + \delta^v_{3,6} - a_6) \\ & + \lambda^c_7 (a_5 + \delta^v_{5,7} - a_7) \end{aligned} \tag{4.13}$$

Since computing arrival times and slacks inside the optimization engine is runtime extensive and not appropriate, we can rely on a few flow conditions to eliminate the negative slack terms slk'_j if, $\lambda'_j + \lambda_j = 1, \forall j \in PO$. Besides, the required times can also be removed from the objective function because they are constant during optimization (OZDAL; BURNS; HU, 2012). We can also eliminate cell and segment arrival times by assuming the same flow conservation derived in Section 3.2.2. These flow conservation implies that the sum of input LMs must be equal to the sum of output LMs. Therefore, cell arrival times are canceled out if $\sum_{i \in \mathcal{I}^c_j} \lambda^c_{i,j} = \sum_{k \in \mathcal{O}^c_j} \lambda^c_{j,k}$. Let $\mathcal{O}^s_j$ denote the set containing the indices to segments or pins connected to

output of $s_j \in \mathcal{S}$. Thus, segment arrival times are also canceled out if $\lambda^s_j = \sum_{k \in \mathcal{O}^s_j} \lambda_k$.

The following equations show the flow conservation conditions derived for the arrival times a_1 to a_7 from the Lagrangian function in Equation (4.13).

$$a_1 \implies \lambda^c_{0,1} = \lambda^s_2 \tag{4.14}$$
$$a_2 \implies \lambda^s_2 = \lambda^s_3 + \lambda^s_4 \tag{4.15}$$
$$a_3 \implies \lambda^s_3 = \lambda^c_6 \tag{4.16}$$
$$a_4 \implies \lambda^s_4 = \lambda^s_5 \tag{4.17}$$
$$a_5 \implies \lambda^s_5 = \lambda^c_7 \tag{4.18}$$
$$a_6 \implies \lambda^c_6 = \lambda_6 \tag{4.19}$$
$$a_7 \implies \lambda^c_7 = \lambda_7 \tag{4.20}$$

Under these flow conservation conditions, it is possible to eliminate the arrival times from the objective funcion, which can be rewritten as follows:

$$L_\lambda : \sum_{c_j \in \mathcal{C}} \Big(\sum_{i \in \mathcal{I}^c_j} \lambda^c_{i,j} (\delta^v_{i,j} + \delta^c_{i,j}) \Big) + \sum_{s_j \in \mathcal{S}} \Big(\sum_{i \in \mathcal{I}^s_j} \lambda^s_j (\delta^v_{i,j} + \delta^s_j) \Big) \tag{4.21}$$

From the simplified Lagrangian function in Equation (4.21), we can conclude that, by minimizing the weighted summation of cell/segment delays and Lagrange Multipliers, the total negative slack metric defined in the original objective function (4.1) is also minimized. Such equation can be minimized using data extracted from the cell library and *.lef*[2] library (KIM et al., 2015), without having to compute arrival times and slacks.

The associated Lagrangian Relaxed Subproblem (LRS) aims to minimize the simplified Lagrangian function L_λ by assigning a layer for each net segment, assuming a set of fixed LMs, as defined in (4.22). Observe that, although we relax the timing modeling constraints, the LRS is still subject to the remaining constraints, as shown in (4.23). From the convex optimization theory it is known that, for any fixed set of LMs, the optimal value of LRS yields a lower bound to the optimal value of the original problem. Since that lower bound depends on the set of LMs, the Lagrange Dual Problem (LDP) aims to maximize

[2]Lef (Library Exchange Format) provides technology information like resistance and capacitance per unit length.

the lower bound from LRS by updating the LMs accordingly (BOYD; VANDENBERGHE, 2004), as defined in (4.24), where Q_λ represents the optimal value from LRS. Therefore, the LRS and LDP problems are solved iteratively.

$$\textbf{LRS} \quad : \quad \min_{\alpha_{j,q}, \forall s_j \in \mathcal{S}} L_\lambda \tag{4.22}$$

$$: \quad \text{s.t. } (4.4), (4.5), \text{ and } (4.11) \tag{4.23}$$

$$\textbf{LDP} \quad : \quad \max_{\lambda \geq 0} Q_\lambda \tag{4.24}$$

The LRS problem specified by Equations (4.22) and (4.23) is an integer programming problem that is solved using a network flow algorithm, as detailed in Section 4.4.1. The LDP problem specified by Equation (4.24) is a convex optimization problem (BOYD; VANDENBERGHE, 2004), but it is non-differentiable (WANG; DAS; ZHOU, 2009). Therefore, it can be solved using the subgradient algorithm, as described in Section 4.4.3.

4.3.3 Obtaining Lagrange Multipliers for Net Segments

During LDP resolution, we rely on slack values computed by the timing analysis engine to update the LMs (as will be detailed in Section 4.4.3). Although industrial timing analyzers report slack values for cell pins and for timing endpoints (OZDAL; BURNS; HU, 2012), timing information for inner net segments is not available to protect their intellectual property algorithms and avoid reverse engineering (HAN et al., 2014). However, we show how to take advantage of the flow conservation conditions (detailed in the previous subsection) to obtain a LM for each net segment, even without slack information. Since interconnections are modeled as RC trees, each segment multiplier can be obtained by back-propagating the LMS from the net sinks to the net source, as illustrated in Figure 41. For example, the multiplier for segment s_2 equals to the sum $\lambda_6 + \lambda_7$, as formalized in the flow conditions derived in Equation (4.15). Also observe that the segment s_2 receives a larger LM (compared to s_4, for instance) due to its impact on both timing paths. Therefore, the optimization engine can take advantage of those larger multipliers to wisely select their layers and reduce timing violations.

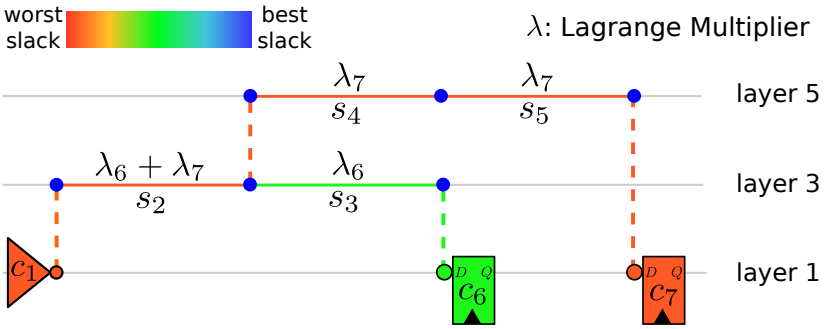


Figure 41: **Obtaining net segment LMs from sink pin LMs.** *Lagrange multipliers are labeled as λ_6 and λ_7 to better illustrate the flow conditions.*

4.4 PROPOSED TECHNIQUE

This section presents the proposed iterative technique to solve the problem formulation from the previous section. First, Section 4.4.1 discusses how to map the proposed instance of LRS problem into a min-cost network flow model and Section 4.4.2 details the adopted cost linearization. Finally, Section 4.4.3 overviews the proposed framework and Section 4.4.4 details the network flow graph generation.

4.4.1 Min-Cost Network Flow Model

Solving the LRS problem as a general integer programming problem may result in prohibitive runtime, especially for large instances. To avoid this overhead, we show that LRS can be interpreted as a transportation problem and then efficiently solved using network flow algorithms. The transportation problem is a classical problem in the network flow theory to which efficient algorithms with theoretical guarantees are available in the literature (BAZARAA; JARVIS; SHERALI, 2011). In the case of the LRS problem defined in Equations (3.28) and (4.23), a single source and a single terminal vertex represent a factory and a warehouse, respectively, while segments and layers can be interpreted as roads with costs and capacities. The impact of assigning a segment to a layer can be interpreted as the transportation cost. Finally, the number of candidate segments for re-assignment corresponds to the total flow to be transported from the source to the terminal vertex. Figure 42 gives a min-cost network flow example for layer as-

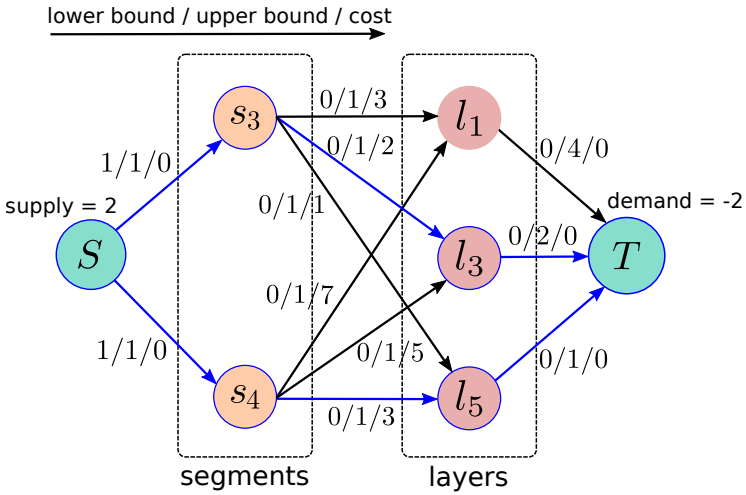


Figure 42: **A min-cost network flow example for layer assignment.** *The min-cost solution assigns s_3 to l_3 and s_4 to l_5.*

signment, where each edge has a lower bound flow, an upper bound flow, and a cost. This example depicts the min-cost flow model for the segments s_3 and s_4 from Fig. 41. Notice that, since s_4 is more critical than s_3, it will be assigned to layer 5 to reduce its delay.

Considering the LRS problem from Section 4.3.2, the binary variable (4.4) is captured through the uni-modularity property inherent from this class of problems (BAZARAA; JARVIS; SHERALI, 2011). The constraint (4.5) can be accounted for by restricting to 1 both the lower and the upper bounds on the flow through the edges connecting the source vertex to the segment vertices. Upper bounds on the edges connecting layers to the terminal vertex capture the capacity constraint (4.11). Unfortunately, the cost function (4.21) presents a few non-linearities that prevent from modeling the cost as a linear function and use efficient state-of-the-art algorithms like network simplex, cost-scaling, and cycle-canceling (KIRÁLY; KOVÁCS, 2012). Therefore, the next section presents several adopted strategies for cost linearization.

4.4.2 Cost Linearization

The delay of a segment s_j on a layer l_q is computed (using Elmore's delay) as $\delta_j^s(q) = R_j^s(q) \cdot (C_j^s(q)/2 + C_j^{down})$, where $R_j^s(q)$ and

$C_j^s(q)$ represent the segment resistance and capacitance on layer l_q, while C_j^{down} refers to s_j's downstream capacitance. The via delay between two consecutive layers l_q and l_{q+1} is computed, similarly to the segment delay, as $\delta^v(q) = R^v(q) \cdot (C^v(q)/2 + C_j^{down})$, where $R^v(q)$ and $C^v(q)$ refer to the via resistance and capacitance, respectively, while C_j^{down} captures its downstream capacitance. Notably, the impact of re-assigning the layer of a given segment s_j to l_q does not restrict to the segment itself. In fact, besides s_j itself, we should also take into account the net driver cell, the upstream segments of s_j, and the downstream net sink cells. Therefore, the impact of re-assigning the layer of a segment is computed as follows.

1) Current s_j's delay and the corresponding via delay from previous segment s_i to s_j itself. Computing the via delay between s_i and s_j requires a binary multiplication (see Equation (4.9)), which introduces a non-linearity in the cost function. Therefore, we employ the following linearization strategy: $\alpha_{i,p} \cdot \alpha_{j,q} \approx \alpha'_{i,p} \cdot \alpha_{j,q}$, where $\alpha'_{i,p}$ corresponds to the layer assigned to segment s_i in the previous iteration. This approximation is reasonable as the proposed technique takes advantage of the iterative nature of an LR-based optimization. Especially in later iterations, when the problem starts to converge, fewer re-assignments are expected to occur and therefore the approximation discrepancy also tends to be reduced. Therefore, the cost of a segment s_j considering its delay and the via delay from its previous segment on layer l_p is computed as in Equation (4.25), where l_q corresponds to s_j's layer.

$$\lambda_j^s \cdot (\delta_j^s(q) + \sum_{k=p}^{q-1} \delta^v(k)), \text{ where } i \in \mathcal{I}_j^s \tag{4.25}$$

2) The downstream capacitance of the net driver cell is affected by the segment capacitance variation, which reacts on cell's delay. Assume that the re-assignment of s_j from layer l_q to l_{q+1} causes a capacitance variation $\Delta C_j = C_j^s(q+1) - C_j^s(q)$. Therefore, we can use a first-order approximation to estimate the impact on driver's delay, as shown in Equation (4.26), where the partial derivative term reflects the cell delay linearization with respect to its downstream capacitance and $\delta_{h,i}^{c'}$ corresponds to the cell's delay in the previous iteration.

$$\lambda_{h,i}^c \cdot (\delta_{h,i}^{c'} + \Delta C_j \cdot \frac{\partial \delta_{h,i}^c}{\partial C_i^{down}}), \forall h \in \mathcal{I}_i^c \tag{4.26}$$

3) The downstream capacitances of upstream segments are also affected by the segment capacitance variation ΔC_j. Let Υ_j be the set of indices to s_j's upstream segments. Therefore, Equation (4.27) accounts for the impact on the delay of each upstream segment, where l_p corresponds to s_i's layer.

$$\lambda_i^s \cdot R_i^s(p) \cdot (C_i^{down} + \Delta C_j), \forall i \in \Upsilon_j \tag{4.27}$$

4) The re-assignment of segment s_j also causes a net slew variation, which in turn impacts on the delay of the sink cells belonging to the downstream path of s_j. Assuming that the re-assignment of segment s_j from layer l_q to l_{q+1} causes a slew variation $\Delta\sigma_j = \sigma_j^s(q+1) - \sigma_j^s(q)$, we can also use a first-order approximation to compute the delay of each sink, as in Equation (4.28). In this equation, the partial derivative term reflects the cell delay linearization with respect to its input slew and $\mathcal{D}_j^s$ denotes the set of indices to downstream sink pins. To compute the slew of each segment on the path from s_j to the net sinks, we adopt the PERI model and Bakoglu's metric (step slew computation) (ZHANG; PAN, 2014), as detailed in Equation (4.29).

$$\lambda_{j,k}^c \cdot (\delta_{j,k}^{c'} + \Delta\sigma_j \cdot \frac{\partial \delta_{j,k}^c}{\partial \sigma_j}), \forall k \in \mathcal{D}_j^s \tag{4.28}$$

$$\sigma_j^s(q) = \sqrt{\sigma_i^2 + (R_j^s(q) \cdot C_j^{down} \cdot \ln 9)^2} \tag{4.29}$$

The Equations (4.25) to (4.29) are then used to compute the re-assignment cost of each segment. The next section details the proposed min-cost flow framework.

4.4.3 Proposed Framework

Figure 43 gives an overview of the proposed incremental layer assignment framework. It receives as input a 3D global routing solution. The incremental layer assignment problem is solved in three major steps.

1) *Update Timing* step first writes a parasitics file in the SPEF format (BHASKER; CHADHA, 2009) containing the distributed RC network information for the circuit. Then *Call Timer* invokes the external timing engine through a Tcl-socket interface (similar to the one described in (KAHNG et al., 2013a)), which returns a report containing

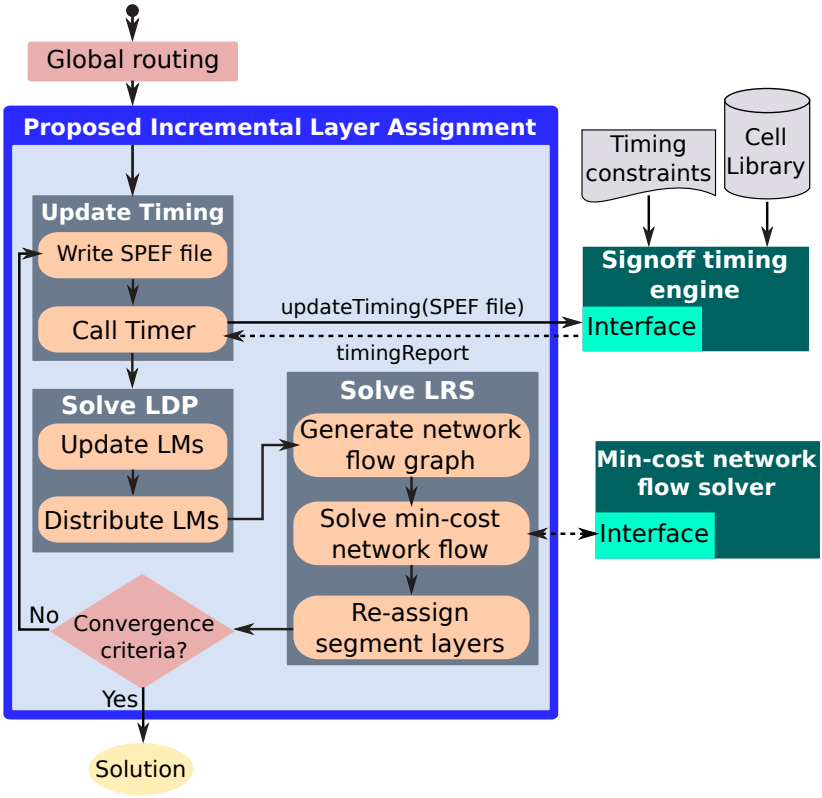


Figure 43: **Overview of the proposed layer assignment framework.**

the slacks for each timing point.

2) *Solve LDP* consists of two sub-steps, as detailed in Algorithm 6. The first one updates LMs so as to increase or decrease their values proportionally to the severity of timing violations measured from the reported slacks. The second sub-step distributes LMs so as to comply with flow conservation conditions. Given the vector $\vec{\lambda}$ of Lagrange Multipliers and the vector $\vec{slk}$ of slacks computed during the timing analysis, Algorithm 6 describes how LMs are updated and distributed. The first sub-step (encapsulated in function UPDATE_LMs) relies on the sub-gradient method (OZDAL; BURNS; HU, 2012) to update LMs for each circuit pin, including cell pins and timing endpoints. The LMs are updated proportionally to the ratio of pin slack to the worst negative slack by visiting each timing endpoint (lines 4 to 6) and every cell timing arc (lines 7 to 11). Our multiplier updating approach

is straightforward and similar to the one adopted in (FLACH et al., 2014). The major difference is that we employ the worst negative slack as a normalization factor, instead of the target clock period. The second sub-step (encapsulated in function DISTRIBUTE_LMs) proportionally distributes the sum of the output LMs of every cell to each of its inputs. The distribution is accomplished by visiting cells in reverse topological order (lines 13-18) (TENNAKOON; SECHEN, 2002), where $\mathcal{O}_j^c$ denotes the set of indices to cells or TEs that are connected to the output of c_j. Finally, LMs are obtained for each net segment (lines 19 to 23), as already exemplified in Fig. 41. The worst-case complexity of the Solve LDP algorithm is $\mathcal{O}(|\mathcal{PO}|+|\mathcal{C}|+|\mathcal{N}|)$.

Algorithm 6: SOLVE_LDP

```
   Input  : λ⃗, slk⃗
   Output: Updated λ⃗
 1 UPDATE_LMs(λ⃗, slk⃗);
 2 DISTRIBUTE_LMs(λ⃗, slk⃗);
 3 Function UPDATE_LMs(λ⃗, slk⃗)
 4     foreach j ∈ TE do
 5         λ_j ← λ_j × (1 + slk_j / WNS);
 6     end
 7     foreach c_j ∈ C do
 8         foreach i ∈ I_j^c do
 9             λ_{i,j}^c ← λ_{i,j}^c × (1 + slk_i / WNS);
10         end
11     end
12 Function DISTRIBUTE_LMs(λ⃗, slk⃗)
13     foreach c_j ∈ C do
14         μ_j ← Σ_{i∈I_j^c} λ_{i,j}^c;
15         foreach i ∈ I_j^c do
16             λ_{i,j}^c ← (λ_{i,j}^c / μ_j) × Σ_{k∈O_j^c} λ_{j,k}^c;
17         end
18     end
19     foreach n ∈ N do
20         foreach s_j ∈ n in reverse topological order do
21             λ_j^s ← Σ_{k∈O_j^s} λ_k^s;
22         end
23     end
```

3) After defining the LMs for each net segment, the *Solve LRS* step aims to solve the problem stated in Equations (3.28) and (4.23). First, it selects critical and non-critical net segments and their respective target layers to generate the network flow graph, as it will be detailed in Section 4.4.4. Then the *Solve min-cost network flow* step

invokes the network flow solver. Among several algorithms available in the literature (BAZARAA; JARVIS; SHERALI, 2011), we employed a cancel-and-tighten implementation of cycle-canceling algorithm, which is very efficient and stable in practice and has a strongly polynomial runtime complexity (KIRÁLY; KOVÁCS, 2012). Finally, *Re-assign segment layers* accomplishes the optimal assignment found by the min-cost flow solver. The three explained steps are repeated until predefined convergence criteria is reached.

4.4.4 Network Flow Graph Generation

For a given global routing, alternative network graphs could be built, depending on which segments are considered timing critical or not. That is why the proposed network flow graph generation relies on two input parameters, num_TE and α, which provide the criteria to select critical and non-critical segments. The first parameter specifies the number of timing endpoints (output pads and inputs of sequential elements) with negative slacks to be used for candidate net selection; the second parameter specifies a threshold factor to help defining whether a net segment should be considered critical or not. The generation of a network graph is performed in three phases: 1) selection of critical segments; 2) insertion of the vertices and edges associated with critical segments (represented by set Γ_c); 3) insertion of vertices and edges associated with non-critical segments (represented by set Γ_{nc}).

Algorithm 7 describes the building of the network flow graph $N(V, E)$ induced by the parameters num_TE and α. The algorithm first creates source and terminal vertices, properly initializes the sets in which critical and non-critical segments will be included, and selects the num_TE timing endpoints with worst slack (lines 1-3), before launching the three-phase procedure. Phase 1 (lines 4-14) selects critical segments by traversing the circuit in reverse topological order, starting at each selected timing endpoint until a timing startpoint is reached. For each visited net, the maximum among the LMs of all its segments (λ^s_{max}) is determined. The algorithm selects as critical segments (line 10) all segments of a given net whose LMs are higher than or equal to the product of the threshold factor by the maximum LM.

Phase 2 (lines 15-28) inserts in the graph the vertices (v_j) representing the critical segments selected by Phase 1, the vertices (v_q) representing candidate layers, and the edges connecting them (v_j, v_q). After a vertex v_j is inserted for each critical segment (line 16), candi-

date layers are selected for it such that all layers below the current one are pruned (line 18). Since the delays of critical segments are expected to be reduced and the benefits from promoting them to upper layers are higher than assigning them to lower layers (because resistance decreases quadratically), the pruning of lower layers reduces the number of edges in the graph without compromising timing improvements. Then the algorithm inserts as many vertices v_q as the number of candidate layers (line 20) and as many edges (v_j, v_q) (line 21). Next, the algorithm identifies (at line 23) the set of segments overlapping with the current segment s_j at a given layer l_q. The algorithm selects as non-critical segments all vertices in that set whose LMs are less than the product of the threshold factor by the LM of the current segment.

Phase 3 (lines 29-39) inserts in the graph the vertices (v_j) representing the non-critical segments selected during Phase 2, the vertices (v_q) representing candidate layers, and the edges connecting them (v_j, v_q). After a vertex v_j is inserted for each non-critical segment (line 30), candidate layers are selected for it such that all layers above the current one are pruned (line 32). Since non-critical segments might release their layers for critical segments, the pruning of upper layers precludes them to use these scarce resources without compromising timing improvements. Besides, candidate layers that would introduce slew violation are also pruned (line 34). Finally, supply and demand attributes are assigned to source and terminal vertices, respectively (lines 40-41).

The time complexity of the proposed algorithm to generate the network flow graph is mainly affected by the number of net segments and the number of layers. Since in the worst-case scenario all the segments (either critical or non-critical) are selected as candidates and the number of layers is a constant term, the asymptotic complexity is $\mathcal{O}(|\mathcal{S}|)$.

Albeit no guarantee can be provided to completely rule out the risk for layer oscillation of timing-critical segments from one iteration to another, the proposed technique relies on the following feature to avoid oscillation: the selection of non-critical candidate segments (Algorithm 7, line 25) is based on the scaled LMs. In other words, a segment is selected as a non-critical candidate in a given iteration if its LM is less than the scaled multiplier of the critical segment. Therefore, if a critical segment becomes non-critical from one iteration to another, it is likely that its new LM value is still sufficiently high so as to prevent it from being selected as a non-critical candidate.

Note that the proposed approach, being an incremental opti-

mization, never lets a segment unassigned, because a segment that is not re-assigned is kept pre-assigned according to the initial legal solution used as a starting point.

Algorithm 7: GEN_NETWORK_FLOW_GRAPH

Input : num_TE, α
Output: Network flow graph $N(V, E)$

Insert the source vertex v_{src} and terminal vertex v_{ter} in V;
$\Gamma_c \leftarrow \emptyset$; $\Gamma_{nc} \leftarrow \emptyset$;
$B \leftarrow num\_TE$ timing endpoints with worst slacks;
foreach $\beta \in B$ **do**
  $pin \leftarrow \beta$
  while $pin \neq$ *timing_startpoint* **do**
    $net \leftarrow$ net connected to pin;
    $\lambda^s_{max} \leftarrow$ highest LM for all segments in the set net;
    foreach *segment* s_j *in the set net* **do**
      if $\lambda^s_j \geq (\alpha \times \lambda^s_{max})$ **then** insert segment s_j in Γ_c;
    end
    $pin \leftarrow$ input pin of net driver cell with worst slack;
  end
end
foreach *critical segment* $s_j \in \Gamma_c$ **do**
  Insert vertex v_j in V;
  Insert edge (v_j, v_{src}) in E with cost=0 and required flow=1;
  foreach *layer* $q = s_j$*'s current layer to* $|\mathcal{L}|$ **do**
    $\kappa \leftarrow$ cost of s_j when assigned to l_q based on (4.25) to (4.28);
    Insert vertex v_q in V;
    Insert edge (v_j, v_q) in E with cost κ;
    Insert edge (v_q, v_{ter}) in E with cost=0 and capacity=$c^e_{j,q}$;
    $O \leftarrow$ segments overlapping with s_j at l_q;
    foreach *segment* s_i *in the set* O **do**
      if $\lambda^s_i < (\alpha \times \lambda^s_j)$ **then** insert segment s_i in Γ_{nc} ;
    end
  end
end
foreach *non-critical segment* $s_j \in \Gamma_{nc}$ **do**
  Insert vertex v_j in V;
  Insert edge (v_j, v_{src}) in E with cost=0 and required flow=1;
  foreach *layer* $q = 1$ *to* s_j*'s current layer* **do**
    $\kappa \leftarrow$ cost of s_j when assigned to l_q based on (4.25) to (4.28);
    if *assigning* s_j *to* l_q *does not introduce slew violation* **then**
      Insert edge (v_j, v_q) in E with cost=κ;
      Insert edge (v_q, v_{ter}) in E cost=0 and capacity=$c^e_{j,q}$;
    end
  end
end
set supply of $v_{src} = |\Gamma_c| + |\Gamma_{nc}|$;
set demand of $v_{ter} = -(|\Gamma_c| + |\Gamma_{nc}|)$;
return $N(V, E)$;

4.5 EXPERIMENTAL VALIDATION

This section presents the experimental evaluation of the proposed technique. The algorithms were implemented in C++ and the min-cost network flow instances were solved using the LEMON library (INITIATIVE, 2016).

Section 4.5.1 details the experimental infrastructure. Section 4.5.2 compares the proposed technique with two state-of-the-art timing-driven incremental layer assignment techniques (YU et al., 2015; LIU et al., 2016) under an industrial timer, while Section 4.5.3 presents an experimental comparison under a simplified timer. Then Section 4.5.4 puts the results into a different perspective to analyze the impact of the timing engine accuracy to guide the optimization. Section 4.5.5 shows how to exploit a hybrid timer to achieve a good tradeoff between runtime and quality. Finally, Section 4.5.6 provides an insightful experimental analysis of the algorithmic decisions.

4.5.1 Infrastructure

Due to the lack of public-domain experimental infrastructures on timing-driven layer assignment, we adapted the ICCAD 2015 Incremental Timing-Driven Contest infrastructure (KIM et al., 2015). That infrastructure was developed considering the Free PDK 45 nm technology library file and unlike the popular ISPD 2008 global routing benchmarks, provides detailed information on cell timing (*Liberty* format) and circuit timing constraints (*SDC* format). This makes the ICCAD 2015 benchmarks appropriate to comparatively evaluate techniques targeting the reduction of timing violations.

We adopted a 9×9 circuit row-height as the size of a G-cell, which is exactly the same used to compute the placement density. We used the 10 metal layers provided in the adopted Free PDK 45 nm library. To compute the number of detailed routing tracks available for each layer, we relied on the metal pitches reported in that library, which also provides resistance and capacitance information for metal layers and vias. After the default grid capacities were set, we adjusted them to account for macro blocks, which are routed in the first 4 metal layers. The global router NCTU-GR (LIU et al., 2013) was invoked to generate the initial 2D routing, while the layer assignment was obtained with the tool NVM (LIU; LI, 2011). All circuits were routed without any edge capacity overflows. It is worth mentioning that any technique

could be used to obtain the initial 3D routing solution such as (WU; DAVOODI; LINDEROTH, 2011; HELD et al., 2015).

The industrial signoff timer Synopsys PrimeTime®, version L-2016.06, was used as golden engine to evaluate the final 3D routing solutions obtained by the techniques under comparison. To invoke the industrial timing engine, four industrial format files were used: 1) *Verilog* containing the circuit description, 2) *SDC* detailing the timing constraints, 3) *Liberty* library describing the cell timing information, and 4) *SPEF* file containing the distributed RC networks.

4.5.2 Comparison under an Industrial Timer

TILA (YU et al., 2015) and CPLA (LIU et al., 2016) are the most recent net-delay-driven approaches and they have the same goal as this work: to improve timing closure. That is why they were selected for a joint comparative evaluation with the proposed technique. The distinct objective functions of the techniques under comparison should be seen as different drivers towards the same ultimate goal. The simpler objective functions employed by TILA and CPLA (i.e. the sum of net segment delay and via delay) actually drive optimization towards better timing. However, we claim that they miss opportunities to reduce the number of violations. We rely on a golden signoff timing engine to provide evidences for supporting our claim. Although the compared techniques solve distinct instances of (essentially) the same general optimization problem, the direct comparison between them allows us to assess to which extent each instance leads to inferior or superior solutions with respect to the ultimate goal of reaching timing closure.

The binaries of TILA and CPLA were obtained with their authors. The experiments were performed on a workstation with a 3.2GHz Intel®i5®CPU with 32GB RAM. To run our technique, we set the parameters $\rho = 20$, $\alpha = 0.001$ for Algorithm 7 and we adopt 8 as the number of iterations of our technique's main loop (Figure 43). The initial values for all the Lagrange Multipliers were set to 1. We also set Synopsys PrimeTime® as the timing engine to be iteratively invoked within our technique.

Table 6 displays the overall circuit characteristics and results. For each circuit, four distinct solutions were evaluated under each metric and labeled as follows: **Initial** corresponds to the initial solutions generated with NCTU-GR and NVM tools. **TILA** (YU et al., 2015) and **CPLA** (LIU et al., 2016) correspond to the results from the net-

Table 6: **Results for the modified ICCAD 2015 Contest benchmarks.** *Comparison with two state-of-the-art techniques (YU et al., 2015; LIU et al., 2016) under the modified infrastructure of the ICCAD 2015 Contest. For the runtime of the proposed technique, we also reported, between parenthesis, how much of the runtime is taken by the industrial timing analyzer.*

Circuit	Solution	TNS (μs)	WNS (ns)	# of T.E. w/ N.S.	# Vias (10^6)	# Vias OVF (10^3)	Runtime (min.)	Crit. / Non. Crit.
superblue18	Initial	-1.73	-6.01	612	1.24	5.87	—	—
Grid: 301x199	TILA	-1.29	-5.05	446	1.34	6.20	1.6	0.5% / 0.5%
Nets: 771K	CPLA	-1.24	-4.80	425	1.29	6.37	15.7	0.5% / 0.5%
Segs: 3.7M	Proposed	-0.61	-3.49	396	1.28	5.93	13.3 (11.1)	0.5% / 1.1%
superblue4	Initial	-4.02	-6.64	1,745	1.61	12.04	—	—
Grid: 369x205	TILA	-3.59	-5.84	1,636	1.70	12.19	2.1	0.5% / 0.5%
Nets: 802K	CPLA	-3.76	-6.29	1,626	1.66	12.59	14.1	0.5% / 0.5%
Segs: 4.6M	Proposed	-1.87	-2.93	1,116	1.70	12.29	15.2 (12.1)	0.3% / 1.9%
superblue16	Initial	-3.08	-5.06	3,064	1.48	4.99	—	—
Grid: 367x199	TILA	-2.24	-4.91	2,462	1.57	5.58	2.2	0.5% / 0.5%
Nets: 999K	CPLA	-2.21	-5.05	2,404	1.54	5.92	24.6	0.5% / 0.5%
Segs: 6.0M	Proposed	-0.20	-2.98	486	1.55	5.30	18.6 (14.7)	0.3% / 1.8%
superblue5	Initial	-1.46	-24.15	410	2.37	23.98	—	—
Grid: 611x282	TILA	-1.48	-18.81	401	2.48	24.55	2.6	0.5% / 0.5%
Nets: 1.1M	CPLA	-1.42	-18.55	389	2.44	24.95	43.0	0.5% / 0.5%
Segs: 6.9M	Proposed	-0.64	-10.27	351	2.42	24.12	20.4 (17.0)	0.4% / 0.6%
superblue1	Initial	-0.45	-4.90	188	2.20	11.98	—	—
Grid: 556x204	TILA	-0.41	-4.78	176	2.32	12.07	3.4	0.5% / 0.5%
Nets: 1.2M	CPLA	-0.41	-4.78	177	2.27	12.92	28.6	0.5% / 0.5%
Segs: 6.2M	Proposed	-0.19	-3.23	131	2.30	12.34	21.5 (17.4)	0.6% / 0.8%
superblue3	Initial	-1.55	-12.89	397	2.31	10.96	—	—
Grid: 632x205	TILA	-1.53	-11.12	376	2.45	11.23	2.4	0.5% / 0.5%
Nets: 1.2M	CPLA	-1.53	-10.93	374	2.37	11.27	30.0	0.5% / 0.5%
Segs: 7.4M	Proposed	-1.40	-8.86	378	2.34	11.12	22.3 (18.7)	0.6% / 0.6%
superblue10	Initial	-33.72	-14.33	6,108	3.52	15.30	—	—
Grid: 538x383	TILA	-31.08	-12.88	6,031	3.69	18.71	5.9	0.5% / 0.5%
Nets: 1.8M	CPLA	-30.75	-12.84	6,057	3.67	21.76	101.7	0.5% / 0.5%
Segs: 13.4M	Proposed	-19.70	-8.62	5,398	3.73	16.20	39.6 (30.6)	0.3% / 1.0%
superblue7	Initial	-2.67	-8.68	1,438	2.92	13.21	—	—
Grid: 394x352	TILA	-1.68	-7.63	1,029	3.15	13.50	4.0	0.5% / 0.5%
Nets: 1.9M	CPLA	-1.47	-6.98	952	3.03	14.13	35.9	0.5% / 0.5%
Segs: 9.05M	Proposed	-0.76	-5.29	816	2.95	13.46	31.2 (26.5)	0.1% / 0.3%
Average red. vs.	TILA	50.20%	35.28%	23.95%	2.39%	2.84%	—	—
Average red. vs.	CPLA	49.10%	34.36%	22.18%	-0.07%	7.41%	—	—

delay-driven techniques. **Proposed** reports the results obtained by the proposed technique. Columns 3 to 5 show the reports from the signoff timer for the following timing metrics: Worst Negative Slack (WNS), Total Negative Slack (TNS), and number of timing endpoints with negative slack. Column 6 reports the number of vias while Column 7 shows the number of vias overflow computed according to the metric detailed in (LIU; LI, 2011). Column 8 displays the runtime in minutes. The last column reports the percentage of critical and non-critical nets selected for each technique for re-assignment. Finally, the two bottom rows detail, for each metric, the average reduction obtained by our technique when compared to TILA and CPLA.

The results concerning TNS and WNS metrics reveal that the proposed technique leads to around 50% and 35% less timing violations than the two related techniques. As a consequence, a reduction of roughly 23% in the number of timing endpoints with violations is observed. Besides the timing violation reductions, the proposed technique requires 2.4% less vias than TILA and roughly the same number of vias as CPLA. In addition, the number of via overflows is 2.8% and 7.4% smaller than TILA and CPLA, respectively. The small overhead in the number of vias required by the proposed technique is an important strength since vias degrade both manufacturing yield and circuit timing (WU; HU; MAHAPATRA, 2005). Therefore, the obtained timing reduction with smaller via penalties puts in evidence that, while focusing on the slowest nets, net-delay-driven techniques might overlook several net segments that truly affect the circuit critical paths. This has two main causes: 1) A timing endpoint with negative slack may not result from a single very slow net, but from a chain of several slow nets and cells; 2) Different setup constraints at different sequential elements have also an important impact on circuit slacks, which is overlooked by a net-delay-driven strategy.

Analyzing the runtime column, one can observe that TILA is roughly 8 times faster than both CPLA and the proposed technique. The longer runtime taken by the CPLA technique was due to the semidefine programming solver, which is much slower than a state-of-the-art network flow solver (employed in TILA) (LIU et al., 2016). On the other hand, the longer runtime taken by the proposed technique was due to the iterative invocation of the signoff timing engine to obtain up-to-date timing reports. As a consequence, the timing analysis takes around 81.7% of the total runtime of the proposed technique. The average runtime for updating the LMs corresponds to 3.9% of the total runtime while the LRS takes 8.5% (including the min-cost flow solver runtime). The remaining 5.9% of the runtime is consumed by parsing and initialization routines. The longer runtime taken by the proposed technique when compared to TILA can be seen as a price to pay to get accurate timing reports to better guide the net segment re-assignments. Nevertheless, the runtime seems reasonable since the proposed technique can optimize circuits with up to 2M nets in less than 40 minutes.

The last column reports the percentage of critical and non-critical nets selected as candidates for re-assignment at each iteration. The two related techniques rely on the same mechanism to select the candidate nets, i.e. 0.5% of critical and 0.5% of non-critical nets are candidates

for re-assignment. Although these ratios can be adjusted, we employed 0.5% to match the ratios reported in the experimental evaluation from (LIU et al., 2016). Recall from Algorithm 7 that the percentage of critical and non-critical nets in the proposed technique is a consequence of parameter ρ. For a fair comparison with related techniques, the parameter $\rho = 20$ was selected to roughly obtain a similar percentage of critical and non-critical selected nets. Nevertheless, Section 4.5.6 analyzes, for different values of ρ, the tradeoff between TNS reduction and runtime obtained by the proposed technique.

4.5.3 Comparison under a Simplified Timer

The improvements shown in the previous subsection are mainly due to our accurate problem formulation and to the use of an industrial timing engine to guide the optimization. To put in evidence the effectiveness of the proposed problem formulation and also isolate the impact of the timing engine, the related techniques should be compared with the proposed technique when the latter is driven by a simplified timer (lumped capacitance and Elmore's delay). Table 7 reports such experimental comparison with TNS and WNS values reported by a simplified timer. From these results, it is possible to conclude that, even when guided and evaluated by a simplified engine, the proposed technique outperforms the related techniques by 31% and 30% under TNS and WNS metrics, respectively. Furthermore, it is worth noting that, when the proposed technique is guided by the simplified timing engine, it obtains the shortest runtimes among the techniques under comparison. These results clearly evidence that the proposed problem formulation is robust enough to use different timing engines.

4.5.4 Impact of the Timing Engine Accuracy

In order to evaluate how important is the use of a signoff engine to guide the optimization (see Figure 43), the proposed technique was run by replacing the industrial timing engine by a simplified (lumped capacitance and Elmore's delay) built-in timing engine. The graphics in Figure 44 compare the TNS and WNS reductions from Table 6 with the ones obtained by our technique when the optimization is guided by the simplified timing engine, referred to as *Ours(simplified)*. Both graphics assume as baseline the reductions obtained by our technique

Table 7: Experimental results under a simplified timer.

Circuit	Solution	TNS (μs)	WNS (ns)	Runtime (min.)
superblue18	TILA (YU et al., 2015)	-1.50	-5.89	1.6
	CPLA (LIU et al., 2016)	-1.44	-5.89	15.7
	Ours	-1.23	-5.14	1.1
superblue4	TILA (YU et al., 2015)	-4.84	-6.43	2.1
	CPLA (LIU et al., 2016)	-5.23	-6.77	14.1
	Ours	-3.36	-4.19	1.2
superblue16	TILA (YU et al., 2015)	-3.89	-5.24	2.2
	CPLA (LIU et al., 2016)	-3.87	-5.17	24.6
	Ours	-0.76	-3.24	1.4
superblue5	TILA (YU et al., 2015)	-6.47	-19.31	2.6
	CPLA (LIU et al., 2016)	-6.12	-19.10	43.0
	Ours	-6.35	-11.03	1.4
superblue1	TILA (YU et al., 2015)	-0.74	-6.69	3.4
	CPLA (LIU et al., 2016)	-0.86	-7.07	28.6
	Ours	-0.53	-5.35	1.8
superblue3	TILA (YU et al., 2015)	-2.15	-12.68	2.4
	CPLA (LIU et al., 2016)	-2.16	-12.68	30.0
	Ours	-1.77	-10.43	1.9
superblue10	TILA (YU et al., 2015)	-36.31	-14.26	5.9
	CPLA (LIU et al., 2016)	-36.17	-14.17	101.7
	Ours	-24.55	-9.54	2.7
superblue7	TILA (YU et al., 2015)	-1.89	-9.90	4.0
	CPLA (LIU et al., 2016)	-1.69	-9.15	35.9
	Ours	-1.14	-5.48	3.0
Average red. vs.	TILA	31.20%	30.52%	—
Average red. vs.	CPLA	30.98%	30.67%	—

when the optimization is guided by the signoff timing engine, referred to as *Ours(industrial)*. The huge pessimism introduced by the simplified engine turned out to guide the technique to optimize paths that are not truly critical, achieving roughly 60% of the TNS reduction obtained when the signoff engine is employed. For some particular cases (e.g. *superblue10* and *superblue7*), that gap is smaller due to the reduced impact of second order effects like resistive shielding. For the WNS metric, the pessimism led to a gap of 20% when different timing engines are used. Besides the importance of using an accurate timing engine, the charts in Figure 44 also put in evidence the robustness of the proposed technique, since it outperforms the related techniques

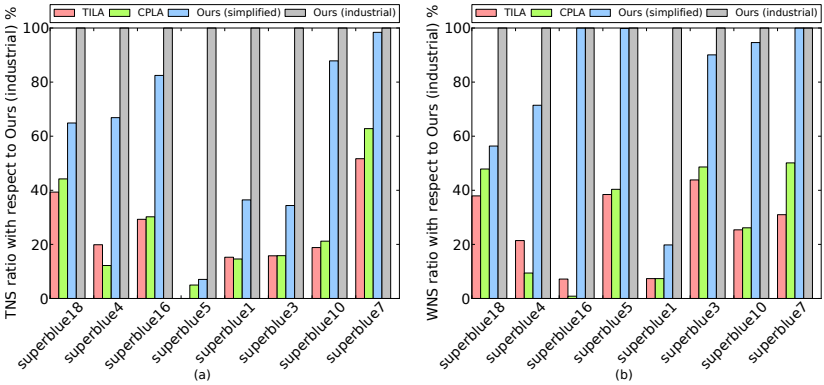


Figure 44: **Impact of the timing engine accuracy.** *Ratio between the timing violation reduction obtained by each technique with respect to our technique using the industrial signoff timer under TNS (a) and WNS (b) metric. Ours (industrial) corresponds to the results reported in Table 6 while Ours (simplified) corresponds to the proposed technique using a simplified timing engine.*

even when a simplified timing engine is used.

4.5.5 Impact of a Hybrid Timer

Although the use of an industrial timer provides more accurate guidance towards timing violation reduction, it leads to a runtime overhead, as shown in Section 4.5.2. Therefore, this section investigates the use of a hybrid timer as an alternative to alleviate the runtime overhead from the use of an industrial timer during the whole optimization flow. We ran an experiment with the proposed technique where the first four iterations invoked the simplified timing engine and the last four iterations invoked the industrial timing engine. Fig.45 (a) and (b) depict the TNS reduction and runtime obtained for three different timing engines: simplified, hybrid and industrial (signoff). Observe that the hybrid engine can obtain reductions very close to those obtained by the industrial engine for 6 out of 8 circuits. In addition, it takes approximately half of the runtime of the industrial timing engine. Therefore, it can be used as an effective alternative to reduce the runtime.

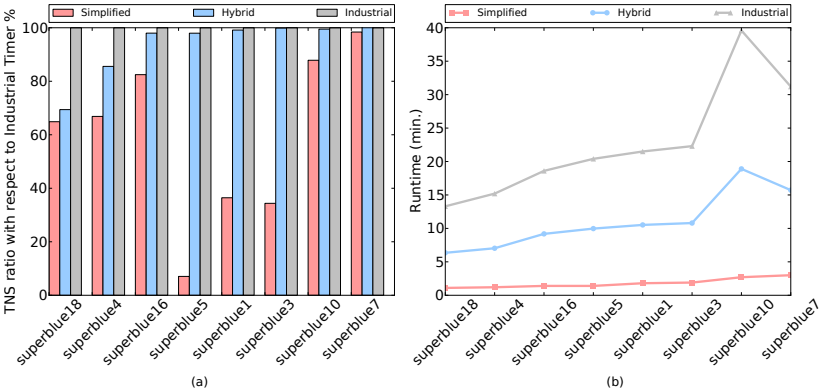


Figure 45: **Impact of a hybrid timer.** *Evaluation under three timing engines: simplified, hybrid and industrial (signoff). (a) TNS reduction for distinct timing engines normalized to an industrial engine and (b) runtime.*

4.5.6 Impact of Algorithmic Decisions

Fig. 46 shows the convergence of the proposed technique under WNS (left y-axis) and TNS (right y-axis) throughout the iterations (iteration ZERO corresponds to the initial solution) for two different circuits. From these charts it is clear that the proposed technique smoothly converges for both WNS and TNS metrics with slight oscillations.

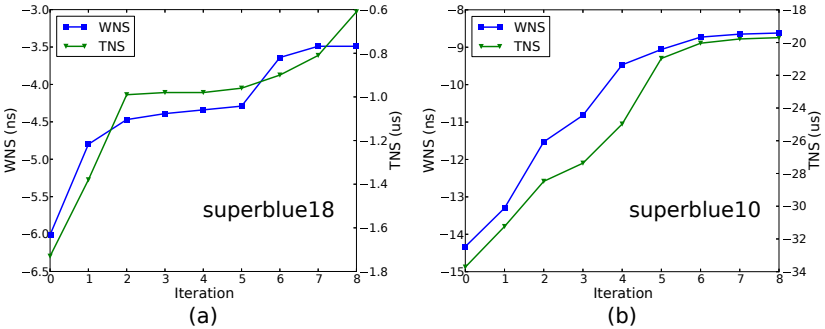


Figure 46: **Algorithm convergence.** *The metrics WNS and TNS along with the number of iterations for superblue18 (a) and superblue10 (b).*

Fig. 47 evaluates the impact of the proposed strategy for pruning lower layer candidates from critical net segments (Algorithm 7, line 18)

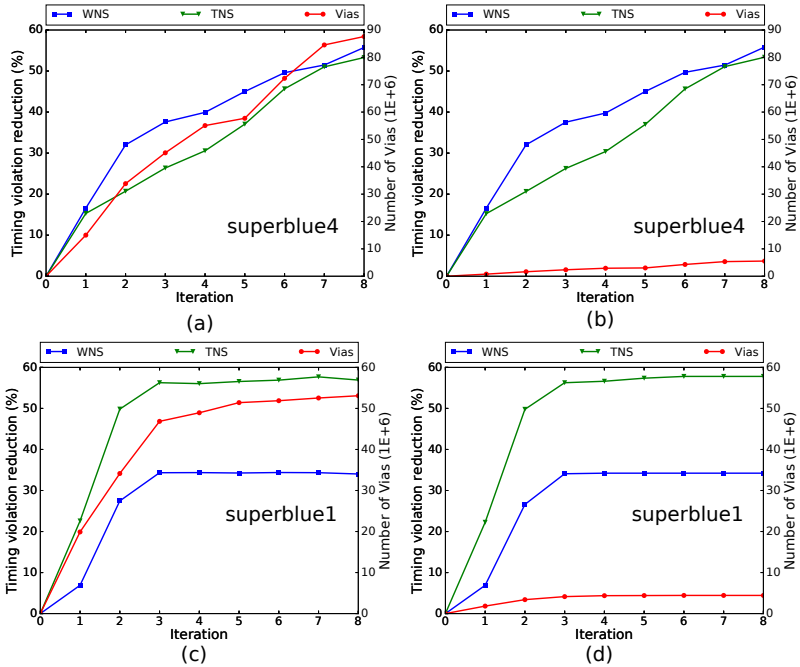


Figure 47: **Impact of the pruning strategy.** *The metrics TNS, WNS, and number of vias are evaluated for circuits superblue4 and superblue1 under two different scenarios. (a) and (b) show the results without pruning while (c) and (d) display the results with pruning.*

and upper layer candidates from non-critical ones (Algorithm 7, line 32). The left side charts give the algorithm behavior for two different circuits when the pruning strategy is not employed whereas the right-hand side charts depict the behavior when it is employed. Observe that for both scenarios, the obtained TNS and WNS reductions are almost the same but the required number of vias is much lower when pruning is performed. The higher number of vias required by the scenario without pruning is mainly due to non-critical segments. Since they can be re-assigned to lower and upper layers to release edges for critical segments, upper layers are generally preferred. However, the assignment of non-critical segments on upper layers considerably increases the number of vias since most of the circuit net segments are routed in lower layers due to their larger capacity. This leads us to conclude that the adopted pruning strategy is effective.

Fig. 48 brings the tradeoff analysis between timing violation reduction (TNS and WNS metrics) and runtime when the number of con-

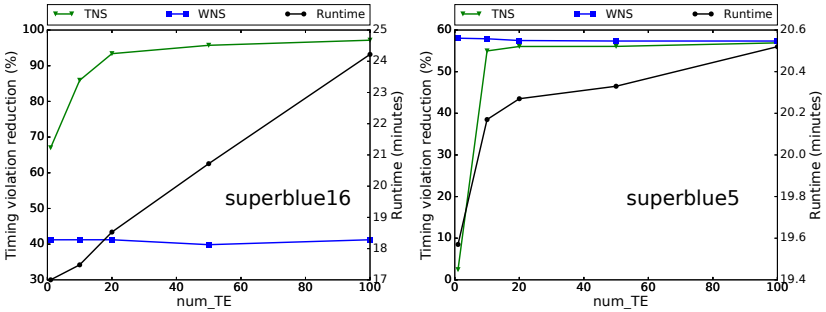


Figure 48: **Algorithm tradeoff between number of TEs and runtime.** *The metrics TNS, WNS, and runtime are evaluated for different number of TEs. (a) Circuit superblue16. (b) Circuit superblue5.*

sidered TEs (i.e. argument *num_TE* from Algorithm 7) is increased. Observe that the proposed technique shows a consistent reduction of TNS when the number of TEs increases, while the runtime increase is roughly linear. The higher number of TEs increases the number of segments for re-assignment which, by its turn, affects the min-cost network flow solving runtime. Also observe that for both charts, the TNS curve becomes flat after a certain number of TEs. This happens because, from that point on, the number of selected critical nets is roughly the same, even though the number of TEs increases. This is a consequence of path sharing among different nets. For both charts, the WNS reduction curves are roughly flat, regardless of the number of TEs, and only small WNS oscillations ($< 1.5\%$) are observed due to the number of near-critical paths.

4.6 CONCLUSIONS

We have proposed a Lagrangian Relaxation formulation that decouples timing-driven incremental layer assignment from the timing engine. The exploitation of flow conservation conditions was the key to enabling the use of an external signoff timing engine. The accurate slack values from the signoff engine provided appropriate guidance towards timing closure, while the linear timing models (for delay and slew) employed for the min-cost flow allowed for fast layer re-assignment. That is why the experimental results showed that the proposed technique can consistently trade a longer runtime for a smaller number of timing violations. As compared to two state-of-the-art methods, the proposed technique led to roughly 50% less timing violation. Besides, the edge

pruning strategy made possible to shrink the problem size while reducing the number of used vias. We also concluded experimentally that the pessimism introduced by a simplified timer guides the optimization engine off critical paths, leaving behind roughly 40% of the room for optimization.

5 OVERALL CONCLUSIONS AND PERSPECTIVES

This chapter summarizes the overall conclusions of this thesis and points out a few future directions.

5.1 CONCLUSIONS

The dramatic impact of interconnects in deep submicron nodes and the fast turnaround time required by contemporary physical synthesis demand effective interconnect synthesis techniques for a successful timing closure. This thesis investigated two important timing optimization problems tied to different steps of the design flow and presented the following innovations.

- A novel Lagrangian relaxation formulation formulation for incremental timing-driven placement that allows the simultaneously handling of late and early timing constraints, while the derived Lagrange multipliers give accurate net-weights that naturally represent the criticality of each net. The use of multipliers as net-weights capture the spatial and temporal criticality of each net, modeling the path sharing and avoiding drastic oscillations. To solve the formulation, a novel discrete search that employs a Euclidean distance to define the neighborhood was devised.
- A new clock-tree-aware ITDP technique that effectively couples IRP and ITDP. Such a joint approach first relocates registers to reduce to clock wiring capacitance and then performs ITDP on combinational and critical sequential cells.
- A novel formulation for incremental timing-driven layer assignment that exploits a few flow conservation conditions to enable the use of industrial signoff timing engines, what is essential for late steps of the design flow. To solve the formulation, a min-cost network flow technique was proposed to handle simultaneously handle critical and non-critical net segments.

The experimental results using benchmark suites derived from industrial circuits showed the effectiveness of the proposed techniques when compared with related works. Such results also reveal that LR is a powerful modeling technique for timing optimization. Since LR allows for decoupling optimization from timing analysis, it is possible to

employ simplified delay models inside the optimization engine while still being guided by accurate slacks computed inside the timing analyzer.

5.2 FUTURE DIRECTIONS

With base on the research reported in this thesis, the following directions for future work can be envisaged:

- Early timing violations are easier to solve when compared to late violations, and the most common strategy to solve is to slow down violating paths (TREVILLYAN et al., 2004). This is generally accomplished by replacing faster cells by slower ones and/or by inserting small buffers. That is why the solving of early violations is generally postponed to the late steps of the physical design flow. Although this work have proposed a timing-driven placement technique that simultaneously handles late and early violations, the impact of such handling in the beginning of the physical design flow was not addressed. Therefore, one possibility of future direction would be to investigate the benefits of addressing early violations right after the placement step.

- Buffer (repeater) insertion is a powerful technique that is intensively used in the physical design flow to (ALPERT et al., 2007) alleviate the delay and slew of long interconnections. Despite its effectiveness, it may lead to excessive area and power overhead when applied alone. That is why a possible future work would be to investigate its integration with timing-driven placement and analyze the benefits of a joint approach in terms of timing, area, and power. One suggested direction would be to reuse the proposed LR formulation for ITDP (Chapter 3) and modify the selection of candidates (lines 43 to 49 in Algorithm 4) to consider the insertion of buffers.

- The cost linearization presented in Section 4.4.2 employed lumped capacitance and Elmore's model, which can be inaccurate, even with Lagrange multipliers updated using a signoff timing engine. One possibility to overcome such an inaccurate modeling would be the use of on-the-fly calibration with the signoff timing engine. The work in (KAHNG et al., 2013b) proposed an offset-based technique to correlate the cell delay and slew models with the values reported by the timing engine. Therefore, the cost linearization presented in Section 4.4.2 could be iteratively improved

to better correlate with the timing results from the signoff timing engine.

- The binary multiplication required to model the dependence between two consecutive segments was approximated using the linearization strategy presented in Section 4.4.2. A possible future direction would be to estimate the impact of such linearization on the solution quality and investigate alternatives for improvements.

- Another possibility of future direction is to introduce a control mechanism to reduce antenna violations in the layer assignment technique. Antenna violations are introduced when long segments are routed in lower layers and connected directly to the gate of the transistor. Conceptually, these long segments can be identified beforehand, at the beginning of each iteration. Therefore, during the network flow graph construction, the segments with long antenna can be constrained to be assigned only to upper layers in order to release the accumulated charges and thus reduce antenna violations. This can be accommodated during the pruning step of the proposed framework from Section 4.4.4.

- Lagrangian relaxation has been shown to be a powerful modeling tool for timing optimization problems such as gate sizing, timing-driven placement, and timing-driven layer assignment. The inherent iterative nature of LR makes it very suitable for this class of problems because it can rely on accurate timing analysis to update the circuit timing information. Although different timing optimization techniques may use LR as modeling engine during the design flow, they are applied as standalone techniques without any kind of feedback between them. Therefore, another possibility of future direction would be a holistic approach for integrating LR at different steps (in cascade), such that the multipliers updated by one technique could be used as a starting point for the upcoming techniques.

- The impact of register placement on heat dissipation is another opportunity for further investigation. Since the relocation of registers towards a more compact clock tree may introduce (or worsen) power hotspots, it is possible to iteratively analyze these hotspots and add extra forces to direct the placement algorithm to alleviate that impact.

- The design rule awareness for advanced technology nodes can be tackled during ITDP by considering minimum space rules between adjancent cells. This would require minor changes in the proposed techniques when evaluating the cell movements.
- The oint evaluation of timing and power closure can be exploited to better analyze the tradeoff between clock interconnect power reduction (provenient from a compacted clock tree) and signal interconnect power penalty.

BIBLIOGRAPHY

AHUJA, R. K.; MAGNANTI, T. L.; ORLIN, J. B. Network flows: Theory, algorithms, and applications. Prentice Hall, 1993. p. 846.

ALPERT, C. et al. Placement: hot or not? In: ACM. *International Conference on Computer-Aided Design*, 2012. p. 283–290.

ALPERT, C. J.; CHU, C.; VILLARRUBIA, P. G. The coming of age of physical synthesis. In: ACM. *International Conference on Computer-aided Design*, 2007. p. 246–249.

ALPERT, C. J. et al. Techniques for fast physical synthesis. *Proceedings of the IEEE*, IEEE, v. 95, n. 3, p. 573–599, 2007.

ALPERT, C. J. et al. What makes a design difficult to route. In: ACM. *International Symposium on Physical Design*, 2010. p. 7–12.

ALPERT, C. J.; MEHTA, D. P.; SAPATNEKAR, S. S. Handbook of algorithms for physical design automation. CRC Press, 2008.

BAZARAA, M.; JARVIS, J.; SHERALI, H. Linear Programming and Network Flows. John Wiley & Sons, 2011.

BAZARAA, M. S.; SHERALI, H. D.; SHETTY, C. M. Nonlinear programming: Theory and algorithms. Wiley-Interscience, 2006. p. 872.

BHASKER, J.; CHADHA, R. Static timing analysis for nanometer designs. Springer, 2009. p. 572.

BOCK, A. et al. Local search algorithms for timing-driven placement under arbitrary delay models. In: ACM. *Design Automation Conference*, 2015. p. 29.

BOYD, S.; VANDENBERGHE, L. Convex optimization. Cambridge University Press, 2004. p. 730.

CHAN, W.-T. J. et al. The itrs mpu and soc system drivers: Calibration and implications for design-based equivalent scaling in the roadmap. In: ACM. *International Conference on Computer-Aided Design*, 2014. p. 153–160.

CHEN, C.-P.; CHU, C. C.; WONG, D. Fast and exact simultaneous gate and wire sizing by lagrangian relaxation. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 18, n. 7, p. 1014–1025, 1999.

CHEON, Y. et al. Power-aware placement. In: ACM. *Design Automation Conference*, 2005. p. 795–800.

CHINNERY, D. High performance and low power design techniques for asic and custom in nanometer technologies. In: ACM. *International Symposium on Physical Design*, 2013. p. 25–32.

CHINNERY, D.; KEUTZER, K. Closing the gap between asic & custom: tools and techniques for high-performance asic design. Springer Science & Business Media, 2002.

CHO, M. et al. BoxRouter 2.0: architecture and implementation of a hybrid and robust global router. In: ACM. *International Conference on Computer-Aided Design*, 2007. p. 503–508.

CHOW, W.-K. et al. Cell density-driven detailed placement with displacement constraint. In: ACM. *International Symposium on Physical Design*, 2014. p. 3–10.

CHOWDHARY, A. et al. How accurately can we model timing in a placement engine? In: ACM. *Design Automation Conference*, 2005. p. 801–806.

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

CONTEST, I. *2015 CAD Contest: Incremental Timing-Driven Placement*. 2015. <http://cad-contest.el.cycu.edu.tw/CAD-contest-at-ICCAD2015/2015_CAD_topic_chair/ProblemC_Myung-Chul.pdf>. Acessado em 12/16/2015.

CORMEN, T. H. et al. Introduction to algorithms. MIT Press, 2009. p. 1312.

COUDERT, O. et al. Incremental cad. In: IEEE. *International Conference on Computer-Aided Design*, 2000. p. 236–244.

COUSSY, P.; MORAWIEC, A. High-level synthesis. Springer, 2010.

DONG, S.; AO, J.; LUO, F. Delay-driven and antenna-aware layer assignment in global routing under multitier interconnect structure. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 34, n. 5, p. 740–752, 2015.

EISENMANN, H.; JOHANNES, F. M. Generic global placement and floorplanning. In: ACM. *Design Automation Conference*, 1998. p. 269–274.

ELMORE, W. The transient response of damped linear networks with particular regard to wideband amplifiers. *Journal of applied physics*, AIP Publishing, v. 19, n. 1, p. 55–63, 1948.

FISHER, M. L. An applications oriented guide to lagrangian relaxation. *Interfaces*, INFORMS, v. 15, n. 2, p. 10–21, 1985.

FLACH, G. et al. Effective method for simultaneous gate sizing and th assignment using lagrangian relaxation. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 33, n. 4, p. 546–557, 2014.

GREG, F. A designer's perspective on timing closure. In: ACM. *Invited Talk at International Symposium on Physical Design*, 2016.

GUTH, C.; LIVRAMENTO, V. et al. Timing-driven placement based on dynamic net-weighting for efficient slack histogram compression. In: ACM. *International Symposium on Physical Design*, 2015. p. 361–366.

GUTHAUS, M. R.; WILKE, G.; REIS, R. Revisiting automated physical synthesis of high-performance clock networks. *Transactions on Design Automation of Electronic Systems*, ACM, v. 18, n. 2, p. 31, 2013.

HALPIN, B.; CHEN, C.; SEHGAL, N. Timing driven placement using physical net constraints. In: ACM. *Design Automation Conference*, 2001. p. 780–783.

HAMADA, T.; CHENG, C.-K.; CHAU, P. M. Prime: a timing-driven placement tool using a piecewise linear resistive network approach. In: ACM. *Design Automation Conference*, 1993. p. 531–536.

HAN, S. et al. A deep learning methodology to proliferate golden signoff timing. In: IEEE. *Design, Automation and Test in Europe*, 2014. p. 260:1–260:6.

HELD, S. et al. Global routing with inherent static timing constraints. In: IEEE. *International Conference on Computer-Aided Design*, 2015. p. 102–109.

HO, R.; MAI, K. W.; HOROWITZ, M. A. The future of wires. *Proceedings of the IEEE*, IEEE, v. 89, n. 4, p. 490–504, 2001.

HU, S.; LI, Z.; ALPERT, C. A polynomial time approximation scheme for timing constrained minimum cost layer assignment. In: ACM. *International Conference on Computer-Aided Design*, 2008. p. 112–115.

HU, S.; LI, Z.; ALPERT, C. A faster approximation scheme for timing driven minimum cost layer assignment. In: ACM. *International Symposium on Physical Design*, 2009. p. 167–174.

HUANG, L. et al. Clock network minimization methodology based on incremental placement. In: ACM. *Asia and South Pacific Design Automation Conference*, 2005. p. 99–102.

INITIATIVE, C.-O. *LEMON: Library for Efficient Modeling and Optimization in Networks*. 2016. <https://lemon.cs.elte.hu>. Acessado em 05/02/2016.

INSIGHTS, I. *Research Bulletin*. 2014. <http://www.icinsights.com/data/articles/documents/737.pdf>. Acessado em 05/02/2015.

ITRS. *International Technology Roadmap for Semiconductors: Interconnect*. 2005. <https://goo.gl/VoZcmv>. Acessado em 04/03/2016.

ITRS. *International Technology Roadmap for Semiconductors: System Drivers*. 2015. <http://www.itrs.net/>. Acessado em 03/03/2015.

KAHNG, A. B. New game, new goal posts: A recent history of timing closure. In: ACM. *Design Automation Conference*, 2015. p. 1–6.

KAHNG, A. B. et al. High-performance gate sizing with a signoff timer. In: ACM. *International Conference on Computer-Aided Design*, 2013. p. 450–457.

KAHNG, A. B. et al. Learning-based approximation of interconnect delay and slew in signoff timing tools. In: IEEE. *International Workshop on System Level Interconnect Prediction*, 2013. p. 1–8.

KAHNG, A. B. et al. Vlsi physical design: From graph partitioning to timing closure. Springer, 2011. p. 310.

KAHNG, A. B.; WANG, Q. An analytic placer for mixed-size placement and timing-driven placement. In: ACM. *International Conference on Computer-Aided Design*, 2004. p. 565–572.

KASHYAP, C. V. et al. Peri: a technique for extending delay and slew metrics to ramp inputs. In: ACM. *International Workshop on Timing issues in the Specification and Synthesis of Digital Systems*, 2002. p. 57–62.

KIM, M. et al. Iccad-2015 cad contest in incremental timing-driven placement and benchmark suite. In: ACM. *International Conference on Computer-Aided Design*, 2015. p. 921–926.

KIM, M.-C.; HU, J.; VISWANATHAN, N. Iccad-2014 cad contest in incremental timing-driven placement and benchmark suite. In: ACM. *International Conference on Computer-Aided Design*, 2014. p. 361–366.

KIM, M.-C. et al. Maple: multilevel adaptive placement for mixed-size designs. In: ACM. *International Symposium on Physical Design*, 2012. p. 193–200.

KIRÁLY, Z.; KOVÁCS, P. Efficient implementations of minimum-cost flow algorithms. *arXiv preprint*, 2012.

KONG, T. T. A novel net weighting algorithm for timing-driven placement. In: ACM. *International Conference on Computer-Aided Design*, 2002. p. 172–176.

KREHER, D. L.; STINSON, D. R. Combinatorial algorithms: Generation, enumeration and search. CRC Press, 1999. p. 329.

LEE, D.-J.; KIM, M.-C.; MARKOV, I. L. Low-power clock trees for cpus. In: ACM. *International Conference on Computer-Aided Design*, 2010. p. 444–451.

LEE, D.-J.; MARKOV, I. L. Obstacle-aware clock-tree shaping during placement. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 31, n. 2, p. 205–216, 2012.

LEE, J.; GUPTA, P. Now Publishers, 2012.

LI, L. et al. An efficient algorithm for library-based cell-type selection in high-performance low-power designs. In: ACM. *International Conference on Computer-Aided Design*, 2012. p. 226–232.

LI, Z. et al. Fast interconnect synthesis with layer assignment. In: ACM. *International Symposium on Physical Design*, 2008. p. 71–77.

LI, Z. et al. Guiding a physical design closure system to produce easier-to-route designs with more predictable timing. In: ACM. *Design Automation Conference*, 2012. p. 465–470.

LIU, D. et al. Incremental layer assignment for critical path timing. In: ACM. *Design Automation Conference*, 2016. p. 85:1–85:6.

LIU, W. et al. NCTU-GR 2.0: multithreaded collision-aware global routing with bounded-length maze routing. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 32, n. 5, p. 709–722, 2013.

LIU, W.; LI, Y. Negotiation-based layer assignment for via count and via overflow minimization. In: ACM. *Asia and South Pacific Design Automation Conference*, 2011. p. 539–544.

LIVRAMENTO, V. et al. Exploiting non-critical steiner tree branches for post-placement timing optimization. In: IEEE. *International Conference on Computer-Aided Design*, 2015. p. 528–535.

LIVRAMENTO, V. et al. Incremental layer assignment driven by an external signoff timing engine. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, p. 1–14, 2017.

LIVRAMENTO, V. et al. Clock-tree-aware incremental timing-driven placement. *Transactions on Design Automation of Electronics Systems*, ACM, v. 21, n. 38, p. 38:1–38:27, 2016.

LIVRAMENTO, V. dos S. *Sizing Discreto Baseado em Relaxação Lagrangeana para Minimização de Leakage em Circuitos Digitais.* Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2013.

LIVRAMENTO, V. S. et al. A hybrid technique for discrete gate sizing based on lagrangian relaxation. *Transactions on Design Automation of Electronic Systems*, ACM, v. 19, n. 4, p. 40, 2014.

LU, Y. et al. Navigating registers in placement for clock network minimization. In: ACM. *Design Automation Conference*, 2005. p. 176–181.

LUO, T.; NEWMARK, D.; PAN, D. Z. A new lp based incremental timing driven placement for high performance designs. In: ACM. *Design Automation Conference*, 2006. p. 1115–1120.

LUO, T. et al. Pyramids: an efficient computational geometry-based approach for timing-driven placement. In: ACM. *International Conference on Computer-Aided Design*, 2008. p. 204–211.

MANOLOPOULOS, Y. et al. R-trees: Theory and applications. Springer Science & Business Media, 2010.

MARKOV, I. L.; HU, J.; KIM, M.-C. Progress and challenges in vlsi placement research. In: ACM. *International Conference on Computer-Aided Design*, 2012. p. 275–282.

MARTIN, G.; BAILEY, B.; PIZIALI, A. Esl design and verification: a prescription for electronic system level methodology. Morgan Kaufmann, 2010.

MARTIN, G. E.; CHANG, H. Winning the soc revolution: Experiences in real design. Springer Science & Business Media, 2003.

MOFFITT, M. D. et al. Path smoothing via discrete optimization. In: ACM. *Design Automation Conference*, 2008. p. 724–727.

NETTO, R.; LIVRAMENTO, V. et al. Evaluating the impact of circuit legalization on incremental optimization techniques. In: IEEE. *Symposium on Integrated Circuits and Systems Design*, 2016. p. 1–1.

NETTO, R.; LIVRAMENTO, V. et al. Speeding up incremental legalization with fast queries to multidimensional trees. In: IEEE. *IEEE Computer Society Annual Symposium on VLSI*, 2016. p. 528–535.

OBERMEIER, B.; JOHANNES, F. M. Quadratic placement using an improved timing model. In: ACM. *Design Automation Conference*, 2004. p. 705–710.

OZDAL, M. M.; BURNS, S.; HU, J. Algorithms for gate sizing and device parameter selection for high-performance designs. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 31, n. 10, p. 1558–1571, 2012.

PAPA, D. et al. Physical synthesis with clock-network optimization for large systems on chips. *Micro, IEEE*, IEEE, v. 31, n. 4, p. 51–62, 2011.

PAPA, D. A. *Broadening the Scope of Multi-Objective Optimizations in Physical Synthesis of Integrated Circuits*. Tese (Doutorado) — The University of Michigan, 2010.

PAPA, D. A. et al. Rumble: an incremental timing-driven physical-synthesis optimization algorithm. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 27, n. 12, p. 2156–2168, 2008.

PILLAGE, L. T.; ROHRER, R. A. Asymptotic waveform evaluation for timing analysis. *Transaction on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 9, n. 4, p. 352–366, 1990.

POPOVYCH, S. et al. Density-aware detailed placement with instant legalization. In: ACM. *Design Automation Conference*, 2014. p. 1–6.

PURI, R.; KUNG, D. S.; DRUMM, A. D. Fast and accurate wire delay estimation for physical synthesis of large asics. In: ACM. *Great Lakes symposium on VLSI*, 2002. p. 30–36.

RABAEY, J.; CHANDRAKASAN, A.; NIKOLIC, B. Digital integrated circuits: a design perspective. Pearson Education, 2003. p. 761.

RAJAGOPAL, K. et al. Timing driven force directed placement with physical net constraints. In: ACM. *International Symposium on Physical Design*, 2003. p. 60–66.

REIMANN, T. J.; SZE, C. C.; REIS, R. Cell selection for high-performance designs in an industrial design flow. In: ACM. *International Symposium on Physical Design*, 2016. p. 65–72.

REN, H. et al. Hippocrates: first-do-no-harm detailed placement. In: ACM. *Asia and South Pacific Design Automation Conference*, 2007. p. 141–146.

REN, H.; PAN, D. Z.; KUNG, D. S. Sensitivity guided net weighting for placement-driven synthesis. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 24, n. 5, p. 711–721, 2005.

RIESS, B. M.; ETTELT, G. G. Speed: Fast and efficient timing driven placement. In: ACM. *International Symposium on Circuits and Systems*, 1995. v. 1, p. 377–380.

SAIT, S. M.; YOUSSEF, H. Vlsi physical design automation: theory and practice. World Scientific, 1999. v. 6.

SAMSUNG. *Exynos 5 Dual*. 2016. <http://www.tweaktown.com/news/25251/samsung_unveils_-exynos_5_dual_the_world_s_first_cortex_a15_soc/index.html>.

SAPATNEKAR, S. Timing. Springer, 2004. p. 306.

SAXENA, P.; LIU, C. Optimization of the maximum delay of global interconnects during layer assignment. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 20, n. 4, p. 503–515, 2001.

SAXENA, P. et al. Repeater scaling and its impact on cad. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 23, n. 4, p. 451–463, 2004.

SHELAR, R. S.; PATYRA, M. Impact of local interconnects on timing and power in a high performance microprocessor. In: ACM. *International Symposium on Physical Design*, 2010. p. 145–152.

SINHA, D. et al. Tau 2013 variation aware timing analysis contest. In: ACM. *International Symposium on Physical Design*, 2013. p. 171–178.

SPINDLER, P. *Efficient Quadratic Placement of VLSI Circuits*. Tese (Doutorado) — Universität München, 2008.

SRINIVASAN, A.; CHAUDHARY, K.; KUH, E. S. Ritual: A performance driven placement algorithm for small cell ics. In: ACM. *International Conference on Computer-Aided Design*, 1991. p. 48–51.

SWARTZ, W.; SECHEN, C. Timing driven placement for large standard cell circuits. In: ACM. *Design Automation Conference*, 1995. p. 211–215.

TENNAKOON, H.; SECHEN, C. Gate sizing using lagrangian relaxation combined with a fast gradient-based pre-processing step. In: ACM. *International Conference on Computer-Aided Design*, 2002. p. 395–402.

TENNAKOON, H.; SECHEN, C. Nonconvex gate delay modeling and delay optimization. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 27, n. 9, p. 1583–1594, 2008.

TREVILLYAN, L. et al. An integrated environment for technology closure of deep-submicron ic designs. *IEEE Design & Test of Computers*, IEEE, v. 21, n. 1, p. 14–22, 2004.

VISWANATHAN, N. et al. Itop: integrating timing optimization within placement. In: ACM. *International Symposium on Physical Design*, 2010. p. 83–90.

WANG, J.; DAS, D.; ZHOU, H. Gate sizing by lagrangian relaxation revisited. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 28, n. 7, p. 1071–1084, 2009.

WANG, L.-T.; CHANG, Y.-W.; CHENG, K.-T. T. Electronic design automation: synthesis, verification, and test. Morgan Kaufmann, 2009.

WANG, Y. et al. Clock-tree aware placement based on dynamic clock-tree building. In: IEEE. *International Symposium on Circuits and Systems*, 2007. p. 2040–2043.

WEI, Y. et al. CATALYST: planning layer directives for effective design closure. In: IEEE. *Design, Automation and Test in Europe*, 2013. p. 1873–1878.

WESTE, N.; HARRIS, D. Cmos vlsi design: a system and circuit perspective. Addison-Wesley, 2010. p. 864.

WU, D.; HU, J.; MAHAPATRA, R. Coupling aware timing optimization and antenna avoidance in layer assignment. In: ACM. *International Symposium on Physical Design*, 2005. p. 20–27.

WU, T.; DAVOODI, A.; LINDEROTH, J. GRIP: Global routing via integer programming. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 30, n. 1, p. 72–84, 2011.

XU, X. et al. Parr: Pin-access planning and regular routing for self-aligned double patterning. *ACM Transactions on Design Automation of Electronic Systems*, ACM, v. 21, n. 3, p. 42, 2016.

YU, B. et al. TILA: Timing-driven incremental layer assignment. In: IEEE. *International Conference on Computer-Aided Design*, 2015. p. 110–117.

ZHANG, Y.; PAN, D. Timing-driven, over-the-block rectilinear steiner tree construction with pre-buffering and slew constraints. In: ACM. *International Symposium on Physical Design*, 2014. p. 29–36.

APPENDIX A – The Incremental Legalization Technique

This appendix details the Incremental legalization technique employed in the algorithms from Section 3.3. The text herein presented reproduces part of the work (NETTO; LIVRAMENTO et al., 2016b) that was also developed in the context of this thesis.

The proposed technique relies on an R-tree (MANOLOPOULOS et al., 2010), a data structure specially tailored to store multidimensional data, such as cell boundaries. Section A.1 presents how the R-tree perform fast spatial operations, while Section A.2 details the proposed technique.

A.1 THE R-TREE DATA STRUCTURE

An R-tree is a non-binary tree that models geometric data in a hierarchical way where each node corresponds to the minimum bounding rectangle (MBR) that contains all its children. The leaf nodes of the tree contain pointers to the stored objects, while non-leaf nodes point to their children. The R-tree structure depends on the order the objects are inserted.

Figure 49 (b) shows an R-tree of degree three created by inserting the cell boundaries in Figure 49 (a) in the order $\boldsymbol{c_1}$, $\boldsymbol{c_2}$, ..., $\boldsymbol{c_8}$. Initially, $\boldsymbol{c_1}$ to $\boldsymbol{c_3}$ are inserted directly into the root, since this is the only node in the empty R-tree. Next, the insertion of $\boldsymbol{c_4}$ requires a node splitting operation, since the root node is already full. The adopted splitting strategy aims to minimize the probability that a query range intersects with both the new nodes MBRs. The strategy begins by creating three new nodes that become the new R-tree leaves. Then $\boldsymbol{c_1}$ to $\boldsymbol{c_3}$ are inserted into the leftmost leaf, while $\boldsymbol{c_4}$ is inserted into the middle leaf. Each insertion is handled by traversing the tree in order to find an appropriate leaf to store the new object. In the given example, the next insertions do not require splitting any node because there is enough room in the nodes to accommodate all remaining objects (cell boundaries). An object is inserted in the leaf that results in minimum MBR area increase. As a consequence, $\boldsymbol{c_5}$ and $\boldsymbol{c_6}$ are inserted in the rightmost leaf, since the MBR area increase would be greater if they were inserted into the middle one. Finally, $\boldsymbol{c_7}$ and $\boldsymbol{c_8}$ are added into the middle leaf, resulting in the R-tree in Figure 49 (b).

Once the R-tree is constructed, a spatial query can be performed by traversing it, while pruning nodes whose MBR do not intersect with the query region. For example, Figure 49 (c) shows the geometric representation of the R-tree from Figure 49 (b). Suppose we want to

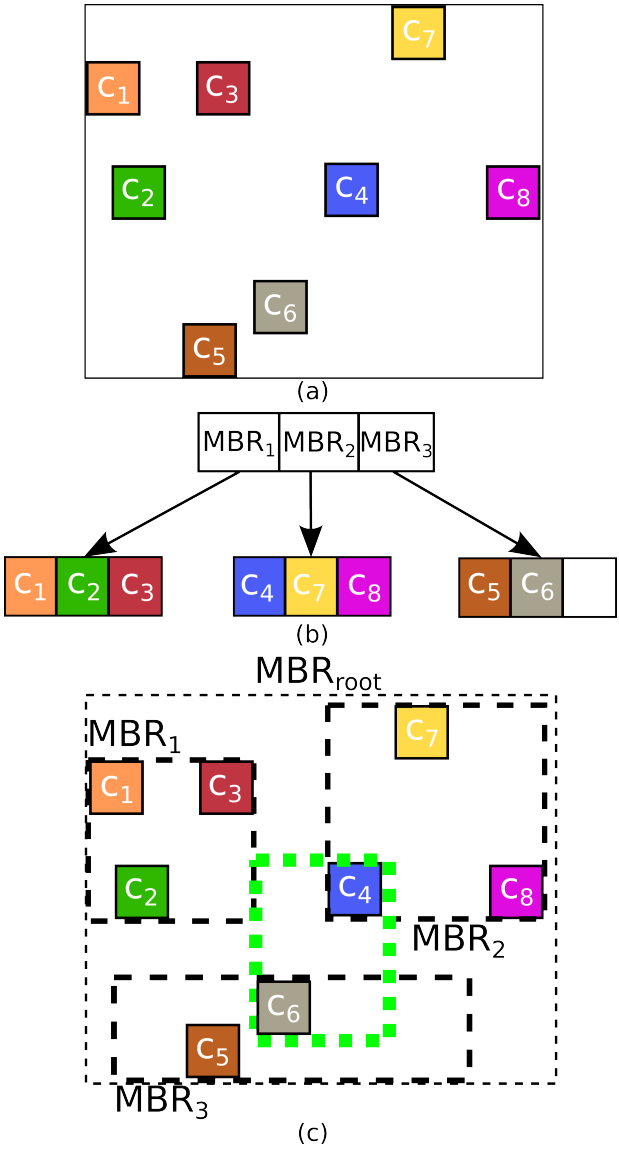


Figure 49: **Example of how an R-tree works.** (a) Example of cell boundaries in a circuit (colored squares). (b) A possible R-tree generated from the data in (a). (c) Geometric representation of the R-tree from (b). The dashed green region represents the range of a spatial query.

find the objects intersecting with the dashed green region in this Figure. First, the R-tree will check in the root node for MBRs intersecting with this region (MBR_2 and MBR_3). Then it visits the leaves and identifies the objects that intersect with the query region (c_4 and c_6 in this case). Observe that the R-tree efficiently reduces the search time by pruning MBRs that do not satisfy the search query. In the example, MBR_1 is pruned and the search does not have to check for intersections with c_1, c_2 and c_3. In this way, the data structure plays the role of a filtering mechanism which performs spatial queries without examining all stored objects.

An object deletion, by its turn, requires tree traversal in order to find the leaf containing the target object to remove. If after the deletion the found leaf contains less than a minimum number of objects, it is also removed from the tree and all its remaining objects are reinserted.

The complexity of R-tree operations is proportional to its height, which is bounded by a balancing algorithm after any object is inserted or removed. The maximum height of the tree is logarithmic with relation to the number of objects. Therefore, the complexity of all R-tree operations (search, insertion and deletion) is $\mathcal{O}(\log(n))$, where n is the number of objects stored in the tree. However, in practice the R-tree height depends on the order the objects were inserted/removed. Efficient packing algorithms can be used to better arrange the data in the tree by inserting all objects at once, improving the runtime of spatial queries.

A.2 THE PROPOSED INCREMENTAL LEGALIZATION

Figure 50 shows how the proposed legalization technique works by means of an example. In Figure 50, the black rectangle represents a macro block, while the colored rectangles are cells. First, our technique divides the circuit in a set of subrows $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_n\}$, such that no subrow overlaps any macroblock. This way the legalization algorithm does not have to handle placement blockages since they are not present in the subrows. Given this set of subrows, it creates a set of $n+1$ R-trees. The first R-tree, denoted as R_Σ contains all subrows. Then for each subrow σ_i there is an R-tree R_{σ_i} containing the cells in that subrow. By storing the circuit cells in R-trees, the proposed technique efficiently identifies cell overlaps by using the R-trees fast spatial queries. In addition, it can update the data structures after each placement transformation by means of R-trees insertion and deletion

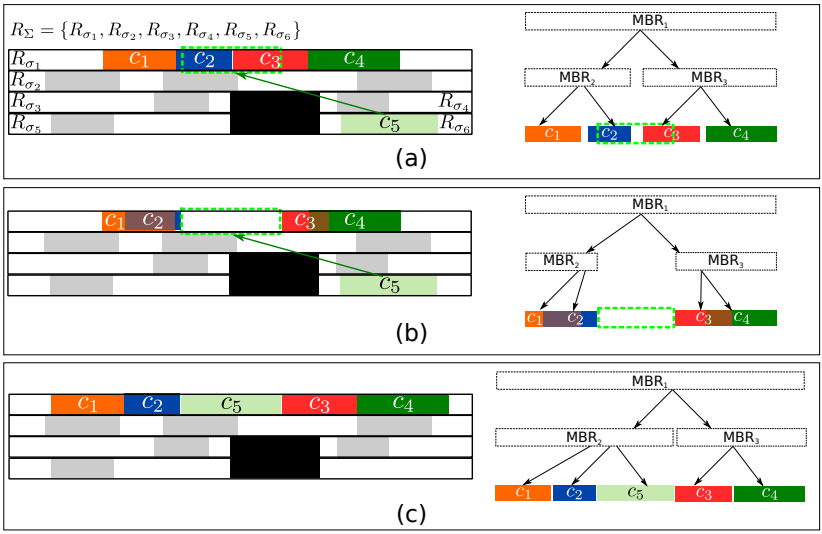


Figure 50: **Example of how the incremental legalization works.** (a) Cell c_5 targeting a new location at row R_{σ_1}. (b) Overlapping cells c_2 and c_3 are shifted to open space for c_5. (c) Overlapping cells c_1 and c_4 are shifted and c_5 is placed at target location.

operations. For each subfigure in Figure 50, the lefthand side shows the placement snapshot during the legalization, and the righthand side shows the R-tree R_{σ_1}.

In this example, c_5 is moved to the position highlighted by the dashed green rectangle, which overlaps with other cells.

Algorithm 8 presents the steps of the proposed technique. Observe that, although the proposed technique is described only for cell relocations, it can be extended for other placement transformations, since an R-tree requires only the cell boundaries to perform its operations.

Given the set of R-trees and a target position (x_i^t, y_i^t) of cell $c_i \in \mathcal{C}$, our technique proceeds as follows: first, it uses R_Σ to find the subrow σ_i containing the target position. After finding the subrow σ_i, the algorithm uses R_{σ_i} to identify the cells overlapping with c_i, storing the overlapping pairs in a set O (line 1). In Figure 50 (a), the target position of c_5 overlaps with c_2 and c_3. Then it removes all overlapping pairs (c_j, c_i) from O (lines 4 and 5) and, for each one, identifies the movement to remove the overlap (lines 6-10). In Figure 50 (b), c_2 must be shifted to the left and c_3 must shifted to the right. If any movement violates the maximum displacement constraint, the legalization fails, and all cells stay in their original positions (lines

Algorithm 8: INCREMENTAL_LEGALIZATION

Input : R-tree of subrows $\boldsymbol{R_\Sigma}$, R-tree of cells $\boldsymbol{R_{\sigma_i}}$ for each subrow $\boldsymbol{\sigma_i}$, and target position $\boldsymbol{(x_i^t, y_i^t)}$ of cell $\boldsymbol{c_i \in \mathcal{C}}$
Output: ***true*** if it is possible to place $\boldsymbol{c_i}$, false otherwise

1 $\boldsymbol{O \leftarrow \{(c_i, c_j) \mid c_j \in R_{\sigma_i} \wedge (x_j \leq x_i^t + w_i) \wedge (x_j + w_j \geq x_i^t)\}}$;
2 $\boldsymbol{M \leftarrow \{(c_i, (x_i^t, y_i^t))\}}$;
3 **while** $\boldsymbol{O \neq \emptyset}$ **do**
4 $\quad \boldsymbol{(c_i, c_j)} \leftarrow$ an overlapping pair from the set $\boldsymbol{O}$;
5 $\quad \boldsymbol{O \leftarrow O - \{(c_i, c_j)\}}$;
6 $\quad$ **if** $\boldsymbol{(x_j + w_j/2) \leq (x_i^t + w_i/2)}$ **then**
7 $\quad\quad \boldsymbol{(x_j^t, y_j^t) \leftarrow (x_i^t - w_j, y_j)}$;
8 $\quad$ **else**
9 $\quad\quad \boldsymbol{(x_j^t, y_j^t) \leftarrow (x_i^t + w_i, y_j)}$;
10 $\quad$ **end**
11 $\quad$ **if** $\boldsymbol{|x_j^t - x_j^0| + |y_j^t - y_j^0| > \Delta_{max}}$ **then**
12 $\quad\quad$ **return** ***false***;
13 $\quad \boldsymbol{O \leftarrow O \cup \{(c_j, c_k) \mid k \in R_{\sigma_i} \wedge (x_k \leq x_j^t + w_j) \wedge (x_k + w_k \geq x_j^t)\}}$;
14 $\quad \boldsymbol{M \leftarrow M \cup \{(c_j, (x_j^t, y_j^t))\}}$;
15 **end**
16 **foreach** $\boldsymbol{(c_j, (x_j^t, y_j^t)) \in M}$ **do**
17 $\quad$ move $\boldsymbol{c_j}$ to $\boldsymbol{(x_j^t, y_j^t)}$ and update subrow $\boldsymbol{R_{\sigma_i}}$ accordingly;
18 **end**
19 **return** ***true***;

11-12). Otherwise, the next overlapping pairs are queried (line 13). Observe that other metrics such as cell movement can be considered along with the maximum displacement to invalidate the legalization. In Figure 50 (b), the new positions of $\boldsymbol{c_2}$ and $\boldsymbol{c_3}$ overlap with $\boldsymbol{c_1}$ and $\boldsymbol{c_4}$, which must be moved as well. Therefore, in Figure 50 (c) $\boldsymbol{c_1}$ must be shifted to the left and $\boldsymbol{c_4}$ must be shifted to the right. When all overlaps are removed, the algorithm iterates through all movements (saved in lines 2 and 14) applying them and updating the R-tree $\boldsymbol{R_{\sigma_i}}$ (lines 16-18). It is important to notice that all movements are applied only if the legalization succeeds. Otherwise, no cell is moved.

Observe that lines 1 and 14 of Algorithm 8 are executed by searching in $\boldsymbol{R_{\sigma_i}}$, while line 18 updates the R-tree by removing the position $\boldsymbol{(x_j, y_j)}$ from $\boldsymbol{R_{\sigma_i}}$ and adding $\boldsymbol{(x_j^t, y_j^t)}$ to it. With the exception of those lines relying on the R-tree, all other operations are done in constant time. Therefore, the runtime of the proposed technique is dominated by the complexity of R-tree operations, which is equal to $\boldsymbol{\mathcal{O}(\log(k))}$, where $\boldsymbol{k}$ is the average number of cells in a subrow. Since in the worst case all cells in a subrow are moved in a legalization, the complexity of the proposed technique is $\boldsymbol{\mathcal{O}(k \log(k))}$. However, in most of the cases only a subset of the cells in a subrow is moved,

leading to an average runtime lower than the worst case.

APPENDIX B – Results Under the ICCAD 2014 Contest Infrastructure

This appendix reports the experimental evaluation of the proposed ITDP technique under the ICCAD 2014 Contest Infrastructure. Firstly, Section B.1 details the adopted infrastructure. Then Section B.2 compares the results of the proposed approach with the best results available from the top three teams involved in the ICCAD 2014 Contest. Finally, Section B.3 brings evidences of the impact on clock-tree compactness. The experimental results detailed in this section were reported in (LIVRAMENTO et al., 2016), when the non-critical cell relocation technique (described in Section 3.3.4.2) was under development. Therefore, the results herein reported comprehend the IRP and the Solving LR steps from the proposed flow (see Figure 28).

B.1 EXPERIMENTAL INFRASTRUCTURE

The ICCAD 2014 Contest infrastructure (KIM; HU; VISWANATHAN, 2014) consists of 7 circuits with sizes between 131k and 959k cells, a set of scripts, and a cell library with linear delay model (KIM; HU; VISWANATHAN, 2014). For each circuit, there is an initial placement solution (generated by a global placer) and two different constraints (short and long) that limit the maximum cell displacement with respect to the initial solution. The contest goal was to reduce the number of late and early timing violations while ensuring that the solution is legal. According to the contest guidelines, a solution is considered legal if: 1) no overlap occurs, 2) standard cells are aligned to rows and sites, 3) macros and fixed cells were not moved, and 4) a maximum displacement constraint is respected. A few metrics are used to evaluate placement solutions. Late and early total negative slacks (TNS), as well as late and early worst negative slacks (WNS) measure timing violations while the density profile is evaluated using the average bin utilization (ABU) metric (KIM et al., 2012). Besides, an overall quality metric is defined (KIM; HU; VISWANATHAN, 2014) as a function of violation reduction (TNS and WNS) and placement density (ABU). The contest guidelines define a relative importance of late over early violations of 5:1. The signal and clock wirelenghts are estimated as Rectilinear Steiner Minimal Trees using FLUTE tool (CHU; WONG, 2008). The clock topology assumes a single buffer (clock source) to drive all registers. We relied on the scripts provided by the ICCAD Contest to evaluate all placement solutions reported in this section.

Table 8 gives an overview of the ICCAD 2014 Contest benchmark suite, detailing the number of standard cells, number of registers,

number of macro blocks, target clock period, target density (maximum is 1), and maximum displacement limits for short and long scenarios.

In the experiments, we adopted the following values for the predefined constants used in the algorithms: $\boldsymbol{max\_it = 5}$, $\boldsymbol{\rho = 5}$, $\boldsymbol{num\_samples = 3}$. We observed experimentally that those values lead to a good tradeoff between solution quality and runtime. To set the bound $\boldsymbol{SWI_{max} = 4\%}$, we adopted a value compatible with the signal wirelength degradation reported in related works (WANG et al., 2007; LEE; MARKOV, 2012). As opposed to most related works on register placement, which disregard density, we defined the bound $\boldsymbol{PDI_{max} = 0.5\%}$ arbitrarily. The placement density increase never reached that limit for any of the tested circuits. To comply with the ICCAD Contest quality metric, wherein late violations are five times more important than early violations, we modified Algorithm 4 (line 21) by weighting 5:1. The initial values for all the Lagrange Multipliers were set to 1.

Table 8: **ICCAD 2014 ITDP Contest benchmark suite.**

Circuit	# of standard cells	# of sequential elements	# of macros	target clock period (ns)	target density	max. displ. limits (μm)
edit_dist	130674	5661	13	5.0	0.75	30/200
matrix_mult	155341	2898	16	4.4	0.70	30/200
vga_lcd	164891	17079	0	4.0	0.70	10/200
b19	219268	6594	0	5.0	0.76	20/200
leon3mp	649191	108839	0	35	0.70	40/400
leon2	794286	149381	0	64	0.70	30/300
netcard	958792	97831	12	42	0.72	50/400

B.2 THE IMPACT OF THE PROPOSED TECHNIQUE

Tables 9 and 10 display the results for short and long displacement constraints, respectively. For each circuit, four distinct placements were evaluated under each metric: the (Initial) solution provided in the Contest infrastructure, the highest quality solution (Best) obtained among the top three teams for that circuit, the solution obtained from our previous work (GUTH; LIVRAMENTO et al., 2015), and the solution obtained from the proposed approach (Proposed). Since no results were presented for the long displacement in Guth, Livramento et al. (2015), Table 10 compares three (instead of four) solutions per circuit. Columns 3-8 show the values obtained for the metrics dis-

Table 9: **Results for the ICCAD 2014 ITDP Contest benchmarks under short displacement constraints.**

		Late		Early				Steiner WL		
Benchmark	**Solution**	**WNS** (***ns***)	**TNS** (***ns***)	**WNS** (***ns***)	**TNS** (***µs***)	**ABU** (10^{-2})	**Quality**	**Clock** (mm)	**Signal** (m)	**Runtime** (min.)
edit_dist	Initial	-0.81	-94.07	-1.45	-3.45	0.00	—	34.1	5.2	—
cells: 131k	Best	-0.53	-14.95	-0.90	-1.78	0.84	1,137.6	34.1	5.3	1.0
regs: 5.6k	Guth et al.	-0.72	-36.95	-0.88	-1.64	0.00	808.7	34.0	5.2	4.9
macros: 13	Proposed	-0.42	-20.70	-0.83	-1.60	0.04	1,172.2	18.9	5.1	2.3
matrix_mult	Initial	-0.44	-2.61	-0.36	-0.11	0.00	—	18.4	3.1	—
cells: 155k	Best	-0.22	-1.00	-0.31	-0.02	1.01	1,032.9	18.3	3.3	1.4
regs: 2.9k	Guth et al.	-0.39	-1.67	-0.29	-0.04	0.00	558.2	18.6	3.1	5.4
macros: 16	Proposed	-0.03	-0.07	-0.24	-0.03	0.03	1,606.4	12.3	3.0	2.7
vga_lcd	Initial	-1.33	-6.39E+02	-4.58	-47.53	1.26	—	92.0	5.7	—
cells: 165k	Best	-0.29	-0.79	-3.13	-31.61	0.94	1,492.0	91.3	5.7	2.3
regs: 17.1k	Guth et al.	-0.16	-0.23	-2.73	-30.23	1.84	1,542.3	92.1	5.7	7.6
macros: 0	Proposed	-0.07	-0.20	-2.69	-28.48	0.64	1,606.3	61.9	5.8	2.5
b19	Initial	-1.16	-15.78	-3.76	-11.49	2.59	—	61.5	4.2	—
cells: 219k	Best	-0.15	-0.31	-1.77	-4.24	3.53	1,580.8	61.5	4.4	1.5
regs: 6.6k	Guth et al.	-0.12	-0.22	-1.83	-4.06	3.09	1,608.5	61.6	4.2	7.0
macros: 0	Proposed	0.00	0.00	-1.88	-5.14	0.79	1,690.4	37.4	4.3	2.9
leon3mp	Initial	-7.61	-2.82E+**4**	-68.51	-3.62E+**3**	0.78	—	504.6	23.7	—
cells: 649k	Best	0.00	0.00	-52.23	-2.73E+**3**	1.31	1,564.7	504.3	23.7	4.2
regs: 108.9k	Guth et al.	0.00	0.00	-47.64	-2.37E+**3**	1.71	1,584.7	504.8	23.8	143.3
macros: 0	Proposed	0.00	0.00	-53.78	-2.83E+**3**	0.00	1,577.5	336.3	24.3	8.5
leon2	Initial	-10.70	-2.00E+**4**	-1.25E+**2**	-1.05E+**4**	2.46	—	661.6	47.2	—
cells: 749k	Best	0.00	0.00	-82.21	-7.79E+**3**	2.46	1,585.6	661.1	47.2	18.1
regs: 149.4k	Guth et al.	0.00	0.00	-80.62	-8.22E+**3**	2.92	1,571.3	660.3	47.3	281.5
macros: 0	Proposed	0.00	0.00	-87.80	-8.12E+**3**	2.59	1,572.7	369.2	49.0	24.5
netcard	Initial	-7.51	-7.55E+**3**	-1.08E+**2**	-7.35E+**3**	1.13	—	506.7	59.0	—
cells: 959k	Best	0.00	0.00	-80.54	-5.78E+**3**	1.13	1,568.4	506.6	59.0	134.1
regs: 97.8k	Guth et al.	0.00	0.00	-75.05	-5.18E+**3**	2.84	1,562.4	506.9	58.9	126.5
macros: 12	Proposed	0.00	0.00	-63.23	-4.52E+**3**	1.13	1,618.5	276.2	60.6	26.5
Average Red. vs Best		40.6%	32.8%	7.0%	-7.0%	0.9%	10.9%	38.8%	0.0%	—
Average Red. vs Guth et al.		41.9%	36.2%	2.1%	-0.4%	1.0%	35.0%	39.1%	-0.6%	—

cussed in Section B.1. Additionally, columns 9 and 10 report the total clock and signal wirelengths. The last column displays the runtimes reported in Kim, Hu and Viswanathan (2014) and Guth, Livramento et al. (2015), as well as the runtimes we measured when running the proposed approach. The two bottom rows show the average reduction obtained by the proposed approach with respect to the best Contest result and to the result obtained from previous work (GUTH; LIVRAMENTO et al., 2015). A negative percentage indicates worsening from the perspective of a given metric. Let us analyze the results according to four distinct aspects: timing closure, signal wirelength and density, clock-tree compactness, and runtime.

Timing Closure: focusing on late constraints under short displacement (Table 9), the proposed approach removes all violations for the four largest circuits, whereas related techniques (Best and Guth, Livramento et al. (2015)) achieve zero violations only for the three largest ones. Besides, a WNS reduction around 40% and a TNS reduction of more than 32% were obtained over the related techniques. Under long displacement (Table 10), the (Best) related technique could not remove all violations for circuits ***edit_dist*** and ***vga_lcd***, while

Table 10: **Results for the ICCAD 2014 ITDP Contest benchmarks under long displacement constraints.**

		Late		Early				Steiner WL		
Benchmark	**Solution**	**WNS** (***ns***)	**TNS** (***ns***)	**WNS** (***ns***)	**TNS** (***µs***)	**ABU** (**10^{-2}**)	**Quality**	**Clock** (mm)	**Signal** (m)	**Runtime** (min.)
edit_dist	Initial	-0.81	-94.07	-1.45	-3.45	0.00	—	34.1	5.2	—
cells: 131k	Best	-0.54	-18.30	-0.84	-1.46	1.45	1,113.1	34.2	5.3	0.8
macros: 13	Proposed	-0.17	-2.94	-0.91	-1.83	0.06	1,491.6	18.7	5.1	6.1
matrix_mult	Initial	-0.44	-2.61	-0.36	-0.11	0.00	—	18.4	3.1	—
cells: 155k	Best	0.00	0.00	-0.28	-0.04	0.00	1,641.8	18.4	3.1	39.1
macros: 16	Proposed	0.00	0.00	-0.21	-0.02	0.02	1,696.6	11.00	3.0	7.9
vga_lcd	Initial	-1.33	-6.39E+**2**	-4.58	-47.53	1.26	—	92.0	5.7	—
cells: 165k	Best	-0.25	-1.76	-3.27	-31.57	2.99	1,473.8	92.0	5.8	1.5
macros: 0	Proposed	0.00	0.00	-2.81	-27.01	0.13	1,643.4	51.9	5.9	7.7
b19	Initial	-1.16	-15.78	-3.76	-11.49	2.66	—	61.5	4.2	—
cells: 219k	Best	0.00	0.00	-1.77	-4.39	3.51	1,661.4	61.5	4.4	1.5
macros: 0	Proposed	0.00	0.00	-2.04	-5.76	0.47	1,680.5	31.5	4.4	6.3
leon3mp	Initial	-7.61	-2.82E+**4**	-68.51	-3.62E+**3**	0.78	—	504.6	23.7	—
cells: 649k	Best	0.00	0.00	-52.00	-2.70E+**3**	1.31	1,564.7	504.3	23.7	4.3
macros: 0	Proposed	0.00	0.00	-48.16	-2.42E+**3**	0.00	1,591.4	302.8	24.6	19.7
leon2	Initial	-10.70	-2.00E+**4**	-124.82	-1.05E+**4**	2.46	—	661.6	47.2	—
cells: 749k	Best	0.00	0.00	-82.21	-7.79E+**3**	2.46	1,585.6	661.4	47.2	19.1
macros: 0	Proposed	0.00	0.00	-79.69	-7.40E+**3**	2.58	1,593.1	368.3	49.0	39.7
netcard	Initial	-7.51	-7.55E+**3**	-108.11	-7.35E+**3**	1.13	—	506.7	59.0	—
cells: 959k	Best	0.00	0.00	-80.54	-5.78E+**3**	1.13	1,568.4	506.6	59.0	136.5
macros: 12	Proposed	0.00	0.00	-74.72	-5.07E+**3**	1.08	1,593.8	271.2	60.6	75.3
Average Red. vs Best		24.0%	26.3%	4.8%	4.1%	1.2%	7.7%	44.1%	-1.0%	—

the proposed approach removed all violations in every circuit, except for ***edit_dist***. Besides, the proposed approach obtained WNS and TNS reductions around 24% and 26% over the (Best) related technique.

When handling early constraints under long displacement (Table 10), the proposed approach obtained WNS and TNS reductions around 4% over the (Best) related technique. Under short displacement (Table 9), WNS reductions around 7% and 2% were obtained over the related techniques, although some worsening in TNS was observed (around 7% and 0.4%, respectively). Such worsening can be explained by the compliance with the Contest weighting guideline, which ends up reducing the importance of early violations in the objective function. That is why this effect shows up under tighter displacement constraints, where there is less room for optimization. The fact of obtaining improvements in the range of 20-40% over related techniques when handling late constraints is a clear evidence of the effectiveness of the approach for timing closure. TNS improvements for early violations in the range of 2-7% are evidences that the approach has an impact on timing closure that is within the same order of magnitude of the importance assigned to them. On the other hand, in designs where their relative importance is raised, we can expect that the proposed approach, due to its orthogonal treatment of both types of constraints, has the potential for impacting timing closure even further.

Clock-tree compactness: Under short displacement constraints, the proposed approach leads to clock wirelength reductions around 39% as compared to the related techniques. Such improvement provides clear evidence that clock-tree awareness broadens the impact of incremental placement far beyond conventional ITDP. Under long constraints, a reduction of 44% was observed. Such higher reduction indicates that the wider cell movements (allowed by more relaxed displacement constraints) are exploited by the proposed approach to further increase clock-tree compactness.

Signal wirelength and density: In the proposed approach, the support to fostering routability consists in imposing bounds on the increase of placement density and signal wirelength. Therefore, the average variation in signal wirelength and in placement density (ABU) can be used to estimate the approach's potential impact on routability (VISWANATHAN et al., 2010). As compared to the related techniques, the proposed approach leads to ABU reductions around 1%, regardless of how tight displacement constraints are (either short or long). Although a slight worsening was observed in signal wirelength, it was kept around 1% of increase. Besides, under short displacement constraints, the proposed approach achieves quality metrics that are 11% and 35% higher than those obtained by the related techniques under comparison. Under long constraints, the quality metric is about 7% higher than the best Contest results.

Runtime: Due to the distinct configurations of the workstations where different techniques were run, no meaningful average reduction in runtime could be computed. Nevertheless, for the largest circuit, the proposed approach took 75 and 26 minutes under long and short displacement constraints, respectively.

It is important to notice that, since the proposed approach solves a more general problem than the related techniques, the longer runtimes observed for smaller circuits under long displacement constraints can be seen as the price to pay for reaching 40% improvement on both clock-tree compactness and timing closure. Indeed, most of the runtime of the proposed approach is spent in the (iterative) detailed placement step during incremental register placement, which is exactly the mechanism allowing the approach to meet placement density and signal wirelength constraints. Therefore, the longer runtimes can be seen as the price to pay for improving signal wirelength and density.

Evidences of an effective ordering: The adopted order for the proposed decomposition assumed that few registers would be touched by ITDP after register placement. For the adopted infrastructure, on

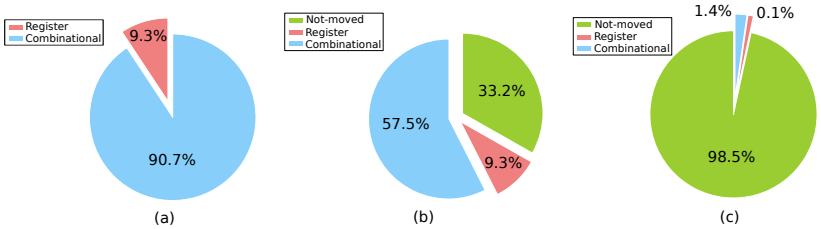


Figure 51: **Statistics on all ICCAD 2014 ITDP Contest's Circuits.** *(a) Percentage of registers and combinational among all standard cells. (b) Percentage of registers, combinational, and non-moved cell after IRP and (c) after ITDP.*

average, 9.3% of the standard cells correspond to registers, as shown Figure 51 (a). The experimental results (under long displacement) revealed that, on average, incremental register placement moves 66.8% of standard cells (57.5% + 9.3%), as shown in Figure 51 (b). Such large number of moved combinational cells is a side effect of the legalization step (since combinational cells must leave space for registers) and of the iterative detailed placement that fosters routability. Figure 51 (c) shows that, since ITDP only relocates cells belonging to the critical paths, it moves 1.5% of the cells and only 0.1% of them are registers. This practical evidence confirms the initial assumption, meaning that the proposed approach adopted adequate order for coupling the subproblems.

Impact of each subproblem: After comparing the results of the proposed joint approach with related works, we illustrate the contribution of each of the coupled techniques. Table 11 shows the intermediate results after incremental register placement (IRP) and the final results after incremental timing-driven placement (ITDP). Note that the intermediate and the final values obtained for WNS and TNS are generally different, especially for the smallest circuits, meaning that IRP left some room for ITDP optimization. However, for the largest circuits, intermediate and final values are often the same, as if IRP had left no room for further optimization. Although diminishing returns should be expected as a result of successive optimization steps, the fact that the largest circuits systematically exhibit such a behavior exposes a known limitation of the Contest infrastructure: it assumes that a single buffer drives the clock interconnect, which adopts a Steiner tree topology. Since a single buffer delivers the clock to all registers (with no intermediate buffers), significant clock skew values are induced. This unrealistic setup (BOCK et al., 2015) shows up in the three largest

Table 11: **Results for the ICCAD 2014 ITDP Contest benchmarks after solving each subproblem.**

		Late		Early				Steiner WL		
Benchmark	**Solution**	**WNS** (*ns*)	**TNS** (*ns*)	**WNS** (*ns*)	**TNS** (*µs*)	**ABU** (10^{-2})	**Quality**	**Clock** (mm)	**Signal** (m)	**Runtime** (min)
edit_dist	IRP	-0.77	-50.68	-0.83	-1.59	0.05	633.6	18.8	5.1	1.8
short	ITDP	-0.42	-20.70	-0.83	-1.60	0.04	1,172.2	18.9	5.1	2.3
matrix_mult	IRP	-0.38	-2.05	-0.23	-0.03	0.06	465.1	12.1	3.0	2.3
short	ITDP	-0.03	-0.07	-0.24	-0.03	0.03	1,606.4	12.3	3.0	2.7
vga_lcd	IRP	-0.28	-1.30	-2.86	-30.83	0.68	1,509.8	61.8	5.8	1.7
short	ITDP	-0.07	-0.20	-2.69	-28.48	0.64	1,606.3	61.9	5.8	2.5
b19	IRP	-0.11	-0.21	-1.88	-5.13	0.81	1,626.9	37.4	4.3	2.6
short	ITDP	0.00	0.00	-1.88	-5.14	0.79	1,690.4	37.4	4.3	2.9
leon3mp	IRP	0.00	0.00	-53.78	-2.83E+**3**	0.00	1,577.5	336.3	24.3	8.5
short	ITDP	0.00	0.00	-53.78	-2.83E+**3**	0.00	1,577.5	336.3	24.3	8.5
leon2	IRP	0.00	0.00	-87.80	-8.12E+**3**	2.59	1,572.7	369.2	49.0	24.5
short	ITDP	0.00	0.00	-87.80	-8.12E+**3**	2.59	1,572.7	369.2	49.0	24.5
netcard	IRP	0.00	0.00	-63.23	-4.52E+**3**	1.13	1,618.5	276.2	60.6	26.5
short	ITDP	0.00	0.00	-63.23	-4.52E+**3**	1.13	1,618.5	276.2	60.6	26.5
edit_dist	IRP	-0.80	-54.79	-0.91	-1.82	0.06	557.2	18.6	5.1	5.8
long	ITDP	-0.17	-2.94	-0.91	-1.83	0.06	1,491.6	18.7	5.1	6.1
matrix_mult	IRP	-0.37	-2.10	-0.21	-0.02	0.35	474.4	10.8	3.0	7.7
long	ITDP	0.00	0.00	-0.21	-0.02	0.02	1,696.6	11.00	3.0	7.9
vga_lcd	IRP	-0.30	-2.23	-2.81	-26.98	0.20	1,523.7	51.7	5.9	3.5
long	ITDP	0.00	0.00	-2.81	-27.01	0.13	1,643.4	51.9	5.9	7.7
b19	IRP	-0.19	-0.44	-2.04	-5.76	0.57	1,564.8	31.5	4.4	5.9
long	ITDP	0.00	0.00	-2.04	-5.76	0.47	1,680.5	31.5	4.4	6.3
leon3mp	IRP	0.00	0.00	-48.16	-2.42E+**3**	0.00	1,591.4	302.8	24.6	19.7
long	ITDP	0.00	0.00	-48.16	-2.42E+**3**	0.00	1,591.4	302.8	24.6	19.7
leon2	IRP	0.00	0.00	-79.69	-7.40E+**3**	2.58	1,593.1	368.3	49.0	39.7
long	ITDP	0.00	0.00	-79.69	-7.40E+**3**	2.58	1,593.1	368.3	49.0	39.7
netcard	IRP	0.00	0.00	-74.72	-5.07E+**3**	1.08	1,593.8	271.2	60.6	75.3
long	ITDP	0.00	0.00	-74.72	-5.07E+**3**	1.08	1,593.8	271.2	60.6	75.3

circuits, where the number of registers is in the order of $\mathbf{10^5}$. Since the application of IRP significantly shrinks the clock interconnect, the consequent reduction in clock skew is so large that the number of timing violations ends up reaching zero for late constraints, as observed for the largest circuits. The reader should be aware that the impact of IRP on the reduction of timing violations is expected to be much smaller for more realistic clock trees. The observed impact should be seen as a side effect (magnified by unrealistic clock trees in the adopted infrastructure) of the very effectiveness of IRP in reducing wirelength. Therefore, although the experimental results are sound as compared to related works (which report results in exactly the same experimental conditions), such unrealistic setup unfortunately prevent us from fully showing the impact of ITDP on large circuits. On the other hand, the impact of such limitation highlights the fact that ITDP is expected to play a major role in more realistic problem instances, such as in the experimental infrastructure adopted in Section 3.4. Therefore, it is fair to expect a higher impact of joint approaches such as the one proposed

in this work.

B.3 FURTHER EVIDENCES OF CLOCK-TREE COMPACTNESS

Figure 52 compares the clock tree routing before and after applying the proposed technique for three circuits from the ICCAD 2014 Contest under long displacement constraints. The circuits are sorted by number of cells to show how the proposed technique behaves for different circuit sizes. To comply with the Contest infrastructure, the clock tree routing was estimated using FLUTE (CHU; WONG, 2008). It can be seen that the proposed technique efficiently compacts the clock tree, achieving up to 48% shorter wirelength. Observe that the impact on routability is quite small since the placement density is reduced for all circuits while the signal wirelength degradations are less than 3%. This is a consequence of the detailed placement step applied after each IRP iteration.

In Section 3.3.3, we reviewed the four main differences between the proposed approach and the most recent technique reported so far for incremental register placement (LEE; MARKOV, 2012). Although a direct comparison with that work was not viable (due to distinct setups and objectives), we performed an extra experiment to quantify the impact of one of their differences (namely the type of forces employed for clock-tree compaction) under exactly the same experimental conditions. Figure 53 compares the joint impact of contraction and clustering forces (as we propose) with the impact of contraction forces only (as proposed in the work (LEE; MARKOV, 2012)). To establish a fair comparison and only for the purpose of this isolated experiment, we modified line 12 in Algorithm 1 so as to capture only the forces between parent and child nodes. Figure 53 presents the result for short and long displacement constraints. It can be seen that the proposed combined strategy achieves more compact clock trees for all circuits regardless of displacement constraints. The combined strategy achieves clock tree wirelength reductions of 5% and 7% for short and long constraints, respectively.

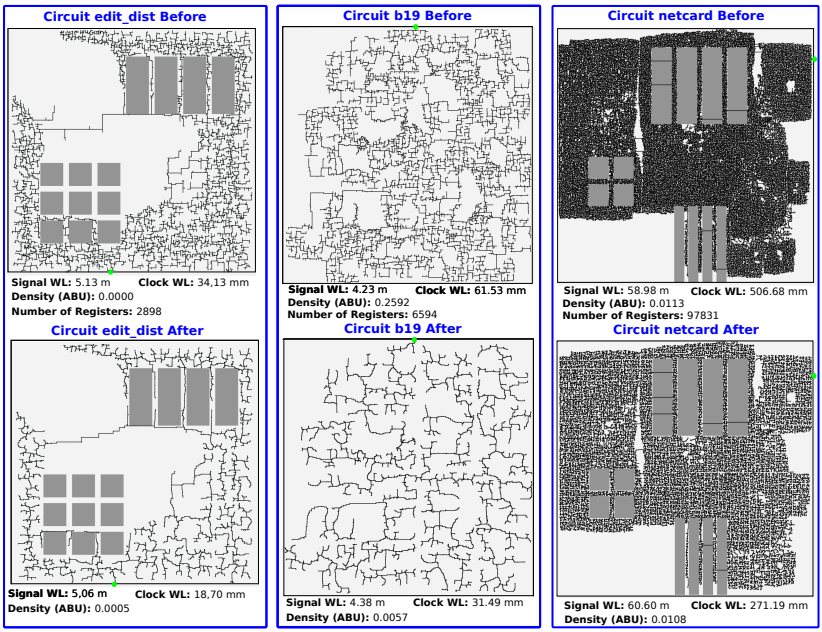


Figure 52: **Snapshots of the clock routing for ICCAD 2014 circuits.** *Results for three circuits from the ICCAD 2014 Contest before (above) and after (below) applying the proposed incremental register placement technique. Green dot at the circuit's boundary represents a clock source. The gray rectangles correspond to macro blocks.*

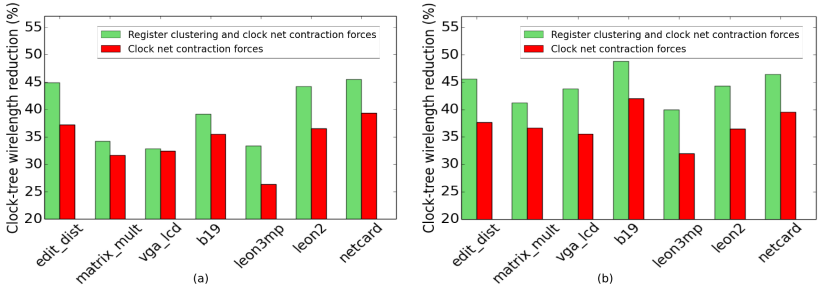


Figure 53: **Clock-tree wirelength reduction.** *Results for short (a) and long (b) constraints obtained by the combination of contraction and clustering forces as compared to the exploitation of contraction forces only.*

APPENDIX C – Results of the Proposed Technique Submitted to the ICCAD 2015 Contest

This appendix presents the official results for the top 3 teams in the ICCAD 2015 Contest, as reported in (CONTEST, 2015). In the technique submitted to the contest, we adopted the following values for the predefined constants used in the algorithms: $\boldsymbol{max\_it = 5}$, $\boldsymbol{\rho = 200}$, $\boldsymbol{num\_samples = 5}$.

Due to some infrastructure issues to handle the clock tree hierarchical distribution, the technique submitted to the Contest did not include the incremental register placement step. Therefore, the final technique encompassed the ITDP techniques, i.e. Solving LR and Non-Critical Cell Relocation Steps from Figure 28. Nevertheless, when evaluated under the ICCAD Contest 2015 infrastructure, the proposed ITDP technique (applied alone) produced the results in Tables 12 and 13, which were awarded the first place in that contest, because they were superior to those obtained by the other 41 competing techniques according to the predefined criteria. These tables report the late/early WNS and TNS results, as well as, density metric (ABU), quality, and runtime. The quality metric aims to cast into a single number the overall improvement including timing violation reduction and density. Equation (C.2) presents how the quality metric is computed from the slack improvement metric (Equation (C.1)) and density variation (i.e. final density minus initial density). Observe that TNS is twice more important than WNS, while late violations are five times more important than early ones.

From the reported results, it is possible to observe that the proposed ITDP technique was able to obtain the best results under the quality metric (column 8 in Tables 12 and 13) for all, except *superblue5* and *superblue10* circuits.

$$\begin{aligned} \text{slack\_imp} &= \mathbf{2} \times (\mathbf{5} \times \boldsymbol{TNS^{L}_{imp}} + \mathbf{1} \times \boldsymbol{TNS^{E}_{imp}}) + \\ &\quad \mathbf{1} \times (\mathbf{5} \times \boldsymbol{WNS^{L}_{imp}} + \mathbf{1} \times \boldsymbol{WNS^{E}_{imp}}) \quad \text{(C.1)} \\ \text{quality} &= \boldsymbol{max}(\text{slack\_imp} \times (\mathbf{1} - \boldsymbol{\Delta}\text{density}), \mathbf{0}) \quad \text{(C.2)} \end{aligned}$$

The final contest evaluation metric, called normalized improvement, aimed to benefit or penalize the quality metric according to the median runtime for the top 5 teams. In other words, the techniques whose runtimes are shorted than the median runtime receive a bonus, while those whose runtimes are longer are penalized. Figure 54 shows two charts containing the normalized improvements for short and long displacement constraints. These results show that the proposed technique consistently outperformed the related techniques. Considering

Table 12: Official Contest Results for top-3 teams under short displacement constraints.

		Late		Early				
Benchmark	**Solution**	**WNS** (***ns***)	**TNS** (***μs***)	**WNS** (***ps***)	**TNS** (***ns***)	**ABU** ($\mathbf{10^{-2}}$)	**Quality**	**Runtime** (min)
superblue18	Initial	-4.55	-1.03	-19.01	-0.28	0.04	—	—
cells: 768k	$\mathbf{1}^{st}$ place	-4.12	-0.94	-3.81	-0.07	0.04	365.3	14.7
macros	$\mathbf{2}^{nd}$ place	-4.39	-1.01	-12.11	-0.14	0.04	179.9	15.9
	$\mathbf{3}^{nd}$ place	-4.15	-1.00	-12.65	-0.07	0.04	257.7	194.0
superblue4	Initial	-6.22	-3.48	-12.55	-0.52	0.04	—	—
cells: 796k	$\mathbf{1}^{st}$ place	-5.94	-3.20	-6.08	-0.17	0.05	287.5	16.7
macros	$\mathbf{2}^{nd}$ place	-6.22	-3.43	-16.17	-0.28	0.05	79.6	30.3
	$\mathbf{3}^{nd}$ place	-6.00	-3.27	-11.67	-0.27	0.04	179.0	718.0
superblue16	Initial	-4.58	-0.78	-10.65	-0.11	0.03	—	—
cells: 982k	$\mathbf{1}^{st}$ place	-4.36	-0.51	-8.38	-0.03	0.04	524.7	19.4
macros	$\mathbf{2}^{nd}$ place	-4.55	-0.67	-3.09	-0.02	0.03	369.6	15.6
	$\mathbf{3}^{nd}$ place	-4.25	-0.73	-0.84	0.00	0.03	386.4	472.4
superblue5	Initial	-25.70	-6.97	-36.77	-0.59	0.02	—	—
cells: 1.09M	$\mathbf{1}^{st}$ place	-25.08	-6.78	-36.77	-0.59	0.02	40.7	22.5
macros	$\mathbf{2}^{nd}$ place	-25.69	-6.94	-32.10	-0.35	0.02	98.0	14.9
	$\mathbf{3}^{nd}$ place	-25.12	-6.90	-30.52	-0.26	0.02	148.2	717.6
superblue1	Initial	-4.98	-0.46	-9.34	-0.32	0.05	—	—
cells: 1.21M	$\mathbf{1}^{st}$ place	-4.67	-0.37	-3.83	-0.04	0.06	447.6	22.8
macros	$\mathbf{2}^{nd}$ place	-4.87	-0.45	-6.74	-0.06	0.06	228.3	80.1
	$\mathbf{3}^{nd}$ place	-4.66	-0.39	-4.21	-0.03	0.05	410.9	720.0
superblue3	Initial	-10.15	-1.50	-78.36	-1.46	0.03	—	—
cells: 1.21M	$\mathbf{1}^{st}$ place	-9.44	-1.37	-65.72	-0.68	0.03	243.2	22.8
macros	$\mathbf{2}^{nd}$ place	-9.86	-1.43	-61.67	-0.90	0.03	159.7	30.2
	$\mathbf{3}^{nd}$ place	-9.71	-1.45	-50.72	-0.56	0.03	214.2	718.4
superblue10	Initial	-16.49	-33.15	-8.62	-0.62	0.04	—	—
cells: 1.88M	$\mathbf{1}^{st}$ place	-16.19	-32.51	-8.62	-0.36	0.04	111.9	40.5
macros	$\mathbf{2}^{nd}$ place	-16.48	-32.84	-6.35	-0.54	0.04	60.6	67.4
	$\mathbf{3}^{nd}$ place	-16.16	-32.99	-5.01	-0.34	0.04	145.9	719.8
superblue7	Initial	-15.22	-1.86	-7.65	-1.99	0.03	—	—
cells: 1.93M	$\mathbf{1}^{st}$ place	-15.22	-1.70	-6.75	-1.94	0.03	98.5	42.8
macros	$\mathbf{2}^{nd}$ place	-15.22	-1.80	-6.93	-1.96	0.03	42.9	35.1
	$\mathbf{3}^{nd}$ place	-15.22	-1.75	-6.93	-1.93	0.03	70.3	579.3

all circuits and both short and long displacement constraints, the obtained improvements over the second place and third place techniques were around 70% and 91%, respectively.

Table 13: Official Contest Results for top-3 teams under long displacement constraints.

		Late		Early				
Benchmark	**Solution**	**WNS** (***ns***)	**TNS** (***μs***)	**WNS** (***ps***)	**TNS** (***ns***)	**ABU** (**10^{-2}**)	**Quality**	**Runtime** (min)
superblue18	Initial	-4.55	-1.03	-19.01	-0.28	0.04	—	—
cells: 768k	**1^{st}** place	-3.82	-0.78	-1.95	-0.01	0.04	613.1	15.9
macros	**2^{nd}** place	-4.23	-0.96	-4.54	-0.04	0.04	355.1	12.1
	3^{nd} place	-3.73	-0.89	-6.01	-0.01	0.04	484.2	675.8
superblue4	Initial	-6.22	-3.48	-12.55	-0.52	0.04	—	—
cells: 796k	**1^{st}** place	-5.76	-2.46	-12.28	-0.05	0.05	507.3	18.6
macros	**2^{nd}** place	-6.22	-3.43	-9.37	-0.08	0.04	208.5	17.8
	3^{nd} place	-6.22	-3.48	-12.55	-0.52	0.04	0.0	720.0
superblue16	Initial	-4.58	-0.78	-10.65	-0.11	0.03	—	—
cells: 982k	**1^{st}** place	-3.85	-0.27	-7.55	-0.04	0.04	894.8	22.4
macros	**2^{nd}** place	-4.53	-0.62	-3.58	-0.04	0.03	394.1	24.0
	3^{nd} place	-3.61	-0.62	-2.83	-0.01	0.03	558.8	719.4
superblue5	Initial	-25.70	-6.97	-36.77	-0.59	0.02	—	—
cells: 1.09M	**1^{st}** place	-24.29	-5.84	-36.77	-0.62	0.02	179.5	25.3
macros	**2^{nd}** place	-25.69	-6.79	-9.40	-0.15	0.02	249.4	19.1
	3^{nd} place	-24.01	-6.71	-15.68	-0.24	0.02	247.3	718.3
superblue1	Initial	-4.98	-0.46	-9.34	-0.32	0.05	—	—
cells: 1.21M	**1^{st}** place	-4.57	-0.35	-16.65	-0.08	0.06	346.6	32.0
macros	**2^{nd}** place	-4.88	-0.45	-9.34	-0.12	0.05	163.7	40.3
	3^{nd} place	-4.98	-0.46	-9.34	-0.32	0.05	0.0	720.0
superblue3	Initial	-10.15	-1.50	-78.36	-1.46	0.03	—	—
cells: 1.21M	**1^{st}** place	-8.71	-1.16	-13.13	-0.21	0.03	551.7	27.0
macros	**2^{nd}** place	-9.27	-1.30	-23.23	-0.17	0.03	427.7	35.7
	3^{nd} place	-9.41	-1.37	-6.40	-0.09	0.03	403.6	720.0
superblue10	Initial	-16.49	-33.15	-8.62	-0.62	0.04	—	—
cells: 1.88M	**1^{st}** place	-16.08	-31.52	-5.15	-0.37	0.04	181.3	37.4
macros	**2^{nd}** place	-16.48	-33.12	-5.01	-0.03	0.04	231.8	48.7
	3^{nd} place	-15.49	-32.92	-6.85	-0.30	0.04	161.8	720.0
superblue7	Initial	-15.22	-1.86	-7.65	-1.99	0.03	—	—
cells: 1.93M	**1^{st}** place	-15.22	-1.51	-6.75	-1.96	0.03	200.7	53.1
macros	**2^{nd}** place	-15.22	-1.80	-6.93	-1.98	0.03	38.6	36.4
	3^{nd} place	-15.22	-1.64	-6.93	-1.94	0.03	129.6	720.0

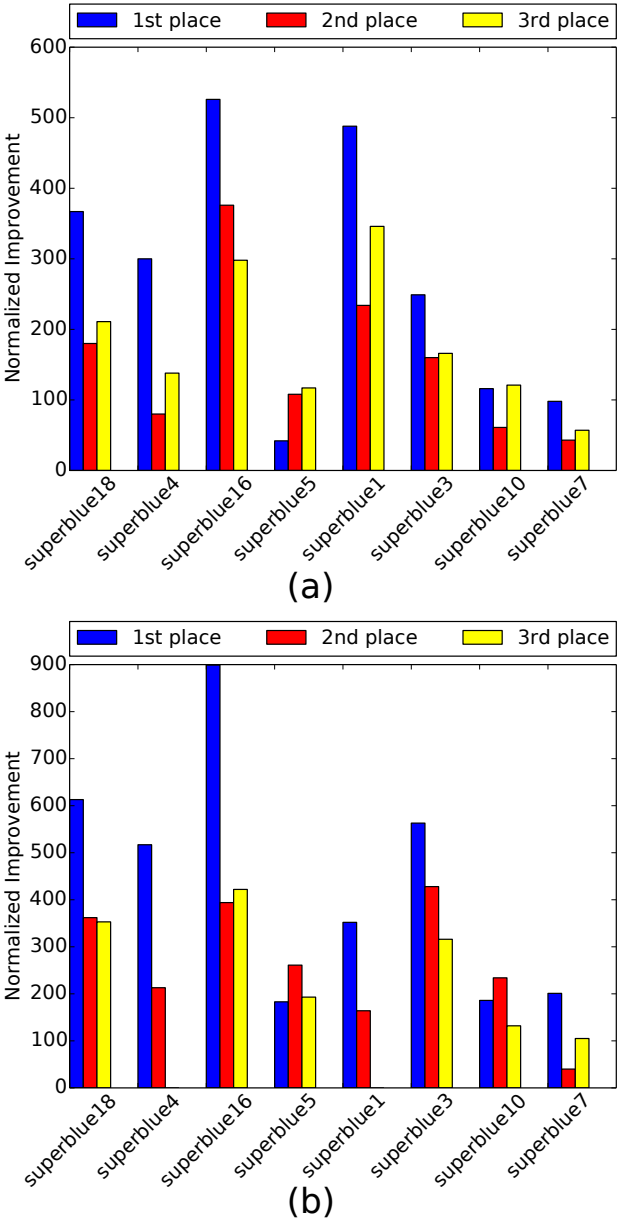


Figure 54: **Normalized improvements for the contest top 3 teams.** The normalized improvement of each circuit includes the quality and runtime. (a) Short and (b) long displacement constraints.